Universidade de Aveiro
2008
Departamento de Electrónica, Telecomunicações e Informática

António Manuel Rodrigues Santos

Modelo de informação para uma aplicação de gestão de redes

Universidade de Aveiro
2008
Departamento de Electrónica, Telecomunicações e Informática

António Manuel Rodrigues Santos

Modelo de informação para uma aplicação de gestão de redes

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Electrónica e de Telecomunicações, realizada sob a orientação científica do Professor Doutor José Luís Oliveira, Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

Para a Celina, Zé Daniel e David.

o júri

presidente

Doutor Atílio Manuel da Silva Gameiro
Professor associado da Universidade de Aveiro

vogais

Doutor Edmundo Heitor Silva Monteiro
Professor associado com Agregação do Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Doutor José Luis Guimarães Oliveira (Orientador)
Professor associado da Universidade de Aveiro

agradecimentos

A realização deste trabalho contou com a colaboração de determinadas pessoas, às quais presto aqui o meu sincero agradecimento.

Gostaria de agradecer ao Engenheiro Jorge Gonçalves pela oportunidade de realizar esta dissertação baseada no trabalho que estou a realizar.

Ao Professor Doutor José Luís Oliveira por ter aceite a orientação deste trabalho.

A todos os colegas e amigos que trabalham comigo, que me ajudaram com ensinamentos, opiniões e trocas de ideias.

palavras-chave NGN, NGOSS, SID, OSS/J, Inventário.

resumo O estado actual das telecomunicações levou a uma mudança no modo como os operadores de telecomunicações fornecem os seus serviços. As operações das operadoras passaram a ser centradas nos serviços a disponibilizar e nos clientes. Esta situação leva a que o tempo dispendido na criação de novos serviços e integração de novas tecnologias tenha de ser o menor possível e com custos sempre decrescentes.
Para resolver estas questões consórcios como o ITU-T e o TMForum criaram normas, nomeadamente o NGN e o NGOSS que tentam a resolver estes problemas.
Este trabalho descreve sucintamente estas normas e apresenta a solução adoptada pela PTInovação para a criação de um módulo de Inventário e modelo de dados usando as normas apresentadas.

keywords NGN, NGOSS, SID, OSS/J, Inventory.

abstract The current state of telecommunications has led to a change in the way. Telecommunications Operators provide their services. The operations of the Telcos began to be service and client centric. This obligates the creation of new services to be cost and time effective.
To address these issues, ITU-T and TMForum created the standards NGN and NGOSS which tend to solve these problems.
This paper describes these standards and presents the solution adopted by PTInovação for the creation of the Inventory module and the data model, using the presented standards.

ÍNDICE

ÍNDICE DE FIGURAS

LISTA DE ACRÓNIMOS

ADSL	Asymetric Digital Subscriber Line
AGORA-NG	Aplicação de Gestão e Operação da Rede de Acesso – Nova Geração
ARPANET	Advanced Research Projects Agency Network
ATM	Asynchronous Transfer Mode
BRI	Basic Rate Interface
BSS	Business Support Systems
CAS	Channel-Association Signaling
CBE	Core Business Entities
CCS	Common Channel Signaling
CVS	Concurrent Versions System
DSL	Digital Subscriber Line
DSLAM	Digital Subscriber Line Access Multiplexer
EFM	Ethernet in the First Mile
EIS	Enterprise Information System
EJB	Enterprise JavaBeans
EMF	Eclipse Modeling Framework
eTOM	enhanced Telecommunication Operations Map
FAB	Fulfillment, Assurance and Billing
FDM	Frequency Division Multiplex
HDSL	High bit rate Digital Subscriber Line
IDE	Integrated Development Environment
IETF	Internet Engineering Task Force
IN	Intelligent Networks
IP	Internet Protocol

ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
J2EE	Java Enterprise Edition
JCP	Java Community Process
JMS	JavaMessage Service
JNDI ENC	Java Naming and Directory Interface Enterprise Naming Context
LAN	Local Area Network
MAN	Metropolitan Area Network
MDA	Model Driven Architecture
MDB	Message Driven Beans
MIB	Management Information Base
N-ISDN	Narrowband Integrated Services Digital Network
NE	Network Element
NGN	Next Generation Networks
NGOSS	New Generation Operations Software and Systems)
OS	Operations System
OSR	Operational Support and Readiness
OSS	Operations Support Systems
OSS/J	OSS through Java
PBX	Private Branch eXchange
PCM	Pulse Code Modulation
PDH	Plesyochronous Digital Hierarchy
POTS	Plain Old Telephone Service
PRI	Primary Rate Interface
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RCS	Revision Control System
RDIS	Rede Digital com Interligação de Serviços

SCP	Signaling Control Points
SDH	Synchronous Digital Hierarchy
SE	Service Element
SHDSL	Single-pair High-Speed Digital Subscriber Line
SID	Shared Information/Data
SNMP	Simple Network Management Protocol
SS7	Signaling System # 7
SSP	Signaling Switching Point
STP	Signaling Transfer Points
TCP/IP	Transmission Control Protocol/Internet Protocol
TDM	Time Division Multiplex
Telco	Telecommunications company
TIP	TMForum Interface Program
TMForum	TeleManagement Forum
TMN	Telecommunication Management Network
TOM	Telecommunication Operations Map
UML	Unified Modeling Language
VDSL	Very high bit rate Digital Subscriber Line
WAN	Wide Area Network
WDM	Wavelength-Division Multiplexing

1. INTRODUÇÃO

1.1. MOTIVAÇÃO

As redes de telecomunicações ocupam hoje um papel muito importante na vida das pessoas. No entanto, estas tecnologias são relativamente recentes e encontram-se em constante evolução. Com o objectivo de oferecer sempre mais e melhores serviços aos clientes as redes de telecomunicações tornam-se cada vez mais complexas, o que faz com que a sua manutenção se torne cada vez mais complexa também.

Para gerirem de forma eficiente estas redes multi-tecnologia, os operadores de telecomunicações evoluíram os seus sistemas de suporte à operação de modo a que estes implementassem os processos operacionais de forma independente das tecnologias de rede.

Estas preocupações levaram a que organismos internacionais (ITU-T, TMForum) desenvolvessem trabalhos na modelação de redes de transporte. Aqui enquadram-se os modelos funcionais (G.805) e os modelos de informação (SID), referências nesta área.

A PT Inovação, como fornecedora de soluções de rede de transporte multi-tecnologia, está a construir um modelo de informação de rede genérico com base nas normas internacionais referidas, para o aplicar na plataforma de gestão AGORA-NG [1].

1.2. OBJECTIVOS

O objectivo desta dissertação consiste em estudar de forma sucinta as tecnologias em questão, construir um gerador automático das entidades do modelo de informação e desenvolver um módulo de inventário, a ser usado dentro das soluções Netb@nd e por aplicações *third-party*.

A modelação será realizada em UML, a implementação do modelo de dados será em BD Oracle e a implementação do modelo de informação será em Java/J2EE com o *Application Server* JBoss.

1.3. ESTRUTURA

Esta dissertação divide-se em duas partes.

Na primeira parte (capítulo 2) são apresentados os conceitos a ser utilizados, a sua evolução e as suas características.

Na segunda parte (capítulo 3) é apresentado o caso de estudo, com as soluções e as ferramentas usadas.

2. CONCEITOS

Neste capítulo serão apresentados os conceitos sobre as tecnologias usadas para a criação do modelo de rede e do modelo de dados, respectivamente a norma ITU-T G.805, o modelo SID da plataforma NGOSS e a iniciativa OSS/J.

Inicia-se o capítulo com uma pequena perspectiva histórica, mostrando as diversas tecnologias que existiram em determinada época e os avanços que se fizeram para chegar aos dias de hoje.

De seguida será apresentado o estado actual dos sistemas de telecomunicações, onde se verão as diversas tecnologias usadas e os desenvolvimentos efectuados para a nova geração de sistemas de telecomunicações.

As Redes de Próxima Geração (NGN – *Next Generation Networks*) serão apresentadas a seguir, descrevendo o conceito, os requisitos para a criação de sistemas NGN e uma apresentação das diversas entidades que estão a trabalhar para a criação de uma solução *standard*.

2.1. BREVE HISTÓRIA

As redes de telecomunicações como as conhecemos hoje são o fruto de uma história com mais de cem anos, que começou com as experiências de Alexander Graham Bell. Por sua vez, a era do computador tem cerca de meio século e tornou-se, nos dias de hoje, na era da Internet. Se no início o desenvolvimento de ambas as tecnologias era relativamente independente, aos poucos as comunicações e a computação foram-se unindo e tornaram-se dependentes uma da outra.

Um ponto de partida para a perspectiva histórica é a década de 1960, com a introdução dos primeiros sistemas digitais na interligação entre centrais de comutação. Esta migração começou por ser lenta devido aos custos elevados da transmissão, mas o aumento do número de utilizadores fez com que o custo operacional baixasse drasticamente. O preço dos equipamentos digitais ia baixando devido à produção em larga escala.

De 1970 a 1980 os desenvolvimentos da comutação de pacotes levou à criação do padrão X.25. No final da década o protocolo TCP/IP foi adoptado como a base da ARPANET, que se tornou na actual Internet.

Nas telecomunicações o aparecimento das PBXs digitais permitiu uma maior eficiência na transferência de chamadas dentro de uma organização [2]. Também na década de 1970 foi apresentado o sistema de sinalização número 7 (SS7), este sistema acrescentou valor à rede telefónica existente, permitindo, por exemplo, a criação do número verde [3].

Nas décadas de 1980 a 2000 o conceito de uma única rede a fornecer o serviço de voz e dados, fez com que aparecesse a primeira rede multi-serviços, baseada no padrão *Narrowband Integrated Services Digital Network* (N-ISDN) [4].

Nesta época as redes locais (LAN) tiveram um rápido crescimento, nomeadamente a *Ethernet* que teve uma enorme aceitação na indústria. As velocidades de débito das LAN aumentaram de 10 Mbps para 100 Mbps em apenas uma década.

A evolução exponencial da Internet deve-se aos avanços nas tecnologias de *routing* e de *switching*. Outro factor que levou a essa evolução foi a melhoria nos mecanismos de transporte, fornecendo um meio eficiente, fiável e económico de comunicação a longa distância e com taxas de débito elevadas. A comutação de pacotes com o modo *Asynchronous Transfer Mode* (ATM) e com velocidades de

155Mbps e a comutação de circuitos *Synchronous Digital Hierarchy* (SDH) com velocidades de 1Gbps.

Nas comunicações foi formulado o conceito de Redes Inteligentes (IN) o que levou à criação de vários serviços de valor acrescentado. Na transmissão o protocolo SDH deu aos operadores a oportunidade de fornecer serviços de transporte rapidamente configuráveis e facilmente geridos, quer para uso próprio quer para o aluguer de circuitos.

Em 1984 havia cerca de 1000 *hosts* na Internet. Foi lançada a *World Wide Web* e o número de *hosts* subiu para um milhão no final de 1992 [5].

Nas redes móveis desde a década de 1980 o desenvolvimento deu-se quer nas redes públicas, quer nas ligações ponto a ponto de alta velocidade, que fornecem o transporte e a interoperabilidade entre redes como uma alternativa às ligações por cabo em zonas remotas.

De entre todos os tipos de redes móveis as que mais se desenvolveram neste período, foram as redes móveis públicas, num curto período de tempo houve 3 gerações de comunicações móveis, que foram classificadas de 1G, 2G, 2,5G e 3G. Cada uma destas gerações incrementa as potencialidades em termos de distância, taxa de transferência, etc. Está em desenvolvimento a quarta geração de comunicações móveis, mas ainda está numa fase inicial.

Outras tecnologias sem fios tornaram-se populares na criação de redes locais. Apareceram vários protocolos com o objectivo de fornecer um meio uniforme de comunicações sem fios, nomeadamente WiFi (IEEE 802.11 a, b, g, etc), WiMax (IEEE 802.16), *Bluetooth*, *Zigbee* (IEEE 802.15.4) entre outros.

Houve desenvolvimentos tecnológicos, mas também de regulação. A separação da empresa Bell System foi o primeiro passo para a liberalização e competição no mundo das telecomunicações, tendência que se espalhou mundialmente.

Durante a década de 1994 a 2003 deu-se o aparecimento dos primeiros fornecedores de acesso privados. Começaram-se a desenvolver normas para a telefonia usando as redes IP e o conceito de uma nova rede multi-serviços foi formulada, a rede de próxima geração. Começaram a ser comercializadas as primeiras licenças da rede móvel de terceira geração (3G), cujo desenvolvimento foi limitado pelas elevadas quantias a que foram vendidas. Durante esta década assistiu-se ao *boom* das empresas tecnológicas, também chamadas dot.com embora, o *crash* de Março de 2000 levasse a que o optimismo ilimitado gerado por estas empresas se desvanecesse. A capacidade de transmissão da fibra óptica aumentou devido ao aumento da velocidade de transmissão e à utilização de múltiplos comprimentos de onda na mesma fibra. Na Internet, o uso das redes IP e as aplicações *web*, tornaram-se uma forma de fornecer aplicações IT (*Information technology*) e comunicações empresariais. A interligação entre as redes de comutação de circuitos e as redes de comutação de pacotes foi possível graças ao desenvolvimento de *media gateways*.

2.2. ACTUALIDADE

Com o conhecimento das capacidades e limitações das tecnologias de rede que estão a ser usadas actualmente, pode-se ter uma melhor perspectiva acerca dos desafios que se põem na passagem para um sistema uniforme de convergência. As tecnologias apresentadas a seguir, serão abordadas de uma forma sucinta devido à vastidão de conceitos que estão inerentes às diversas tecnologias.

2.2.1. REDE COM FIOS

As redes com fios, como o nome indica usam o meio físico como meio de transporte. Estas redes incluem os sistemas tradicionais de voz (PSTN) e dados (tecnologias LAN, tais como *Ethernet*, ATM ou DSL).

O sistema telefónico tradicional (PSTN – *Public Switched Telephone Network*) é um ambiente de comutação de circuitos, que permite que as comunicações de voz e de dados sejam transportados por um par de fios cobre. A rede [Figura 1] é composta por vários elementos, entre eles o terminal do assinante (telefone), o lacete local (troço dedicado que liga o terminal ao comutador – par de fios cobre), os comutadores, que se dividem em comutadores terminais (responsáveis por estabelecer as ligações e estão directamente ligados aos assinantes) e os comutadores de trânsito (interligam comutadores de transito de níveis inferiores ou comutadores terminais) e os troços de interligação (ligações de alta capacidade que transportam a voz – digital e os dados entre comutadores).

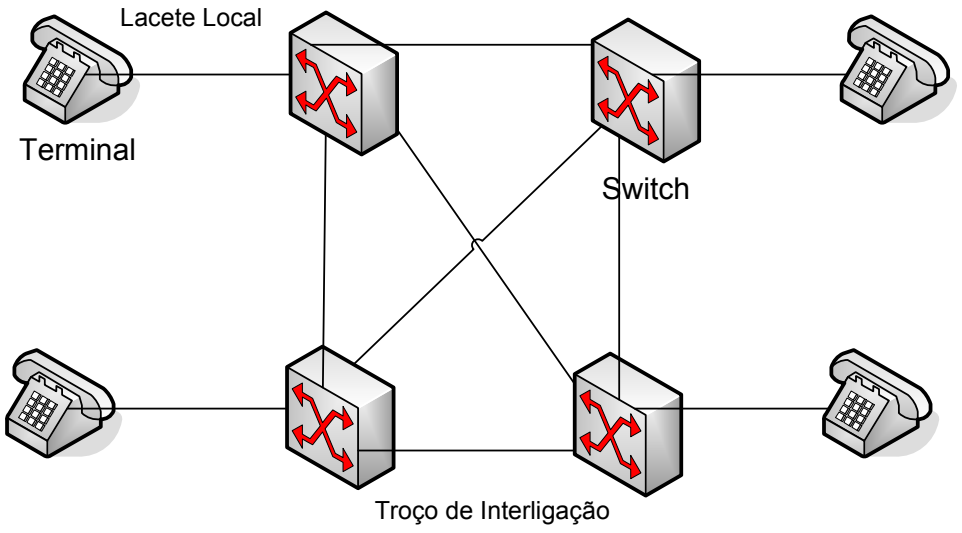


Figura 1 – Elementos do PSTN

A arquitectura da rede telefónica é uma estrela hierárquica com vários níveis [Figura 2], encontrando-se no topo o comutador responsável pelas ligações internacionais, os diferentes níveis dependem da complexidade da rede e do número de assinantes.

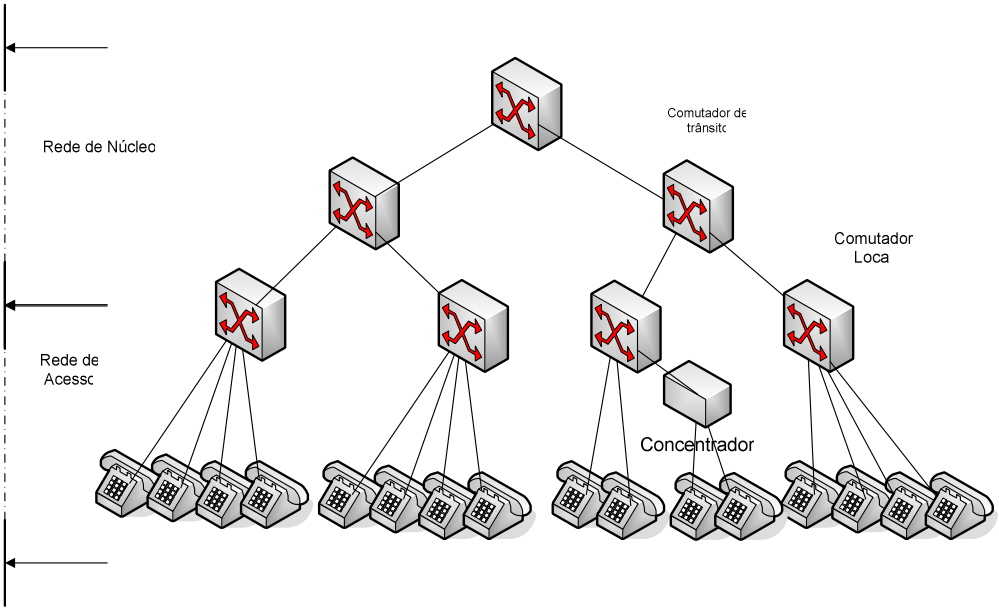


Figura 2 – Estrutura hierárquica da rede telefónica

A sinalização é uma parte muito importante da rede telefónica, a sinalização *in-band* é usada para transmitir informação, como por exemplo o número de telefone ao comutador. Existem mais dois tipos de sinalização: o *Channel-Association Signaling* (CAS) e o *Common Channel Signaling* (CCS). O CAS usa uma linha diferente para a transmissão de sinalização, enquanto que o CCS usa uma rede de comunicações diferente para a troca de informação [3]. O sistema de sinalização nº 7 (SS7) é usado para a rede CCS. O SS7 é composto por três elementos principais: o *Signaling Switching Point* (SSP), que são comutadores telefónicos com aplicações SS7 e terminações de sinalização; *Signaling Transfer Points* (STP), são comutadores de pacotes da rede SS7 que recebem a sinalização e passam-na ao devido destinatário; *Signaling Control Points* (SCP) base de dados que fornece a informação necessária para o processamento de funcionalidades, tais como número verde. [Figura 3].

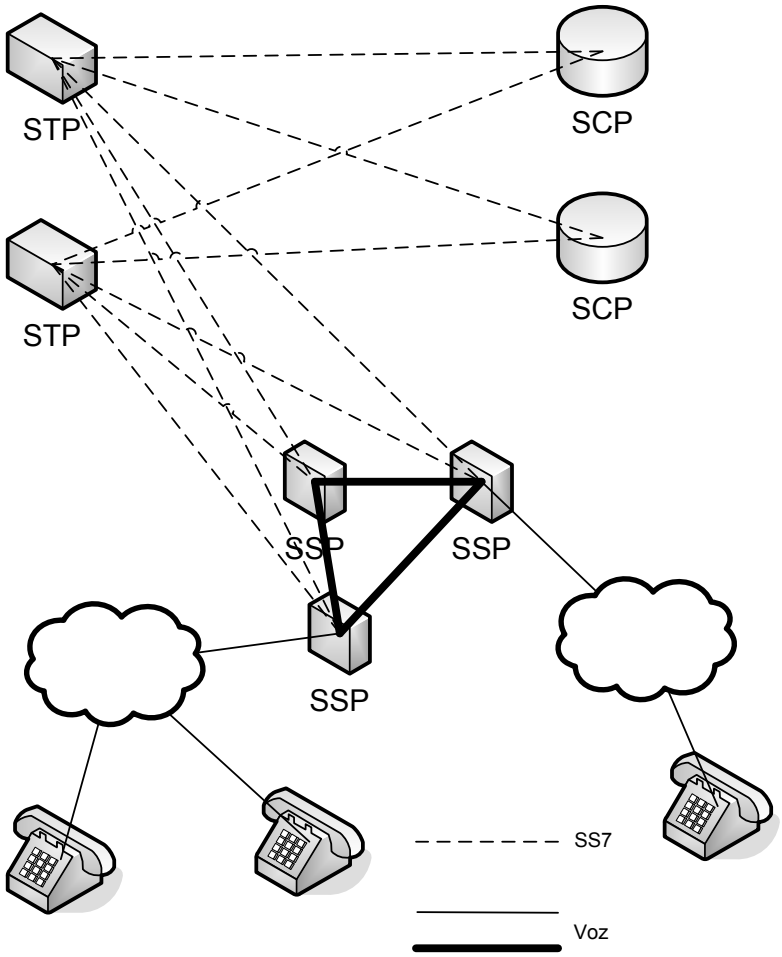


Figura 3 – Rede SS7 simplificada

O transporte de dados na rede PSTN é feito da seguinte maneira: através da rede telefónica analógica, conhecida por *Plain Old Telephone Service* (POTS); através da rede telefónica digital – Rede Digital com Interligação de Serviços (RDIS); através da rede de pares simétricos (DSL).

Os serviços de dados nos sistemas POTS usavam modems que operavam na banda de voz. A largura de banda é restrita, permitindo sinais com largura de banda entre os 300Hz e os 3000Hz, que correspondem a taxas de transmissão entre os 300bps e os 56kbps. Os modems seguem a norma da série ITU-T V. Em 2000 foi lançado a norma V.92 que permite a transferência assimétrica de dados com taxas de 56kbps de *download* e restringe os *uploads* a 48kbps.

O RDIS apareceu como uma evolução natural ao sistema analógico. Era um sistema totalmente digital. O sistema RDIS fornecia dois formatos de interface com o assinante: o *Basic Rate Interface* (BRI) e o *Primary Rate Interface* (PRI) [Figura 4]. A interface BRI era vocacionada para clientes residenciais, seguia a norma ITU-T I.430, fornecendo dois canais de 64kbps cada, chamados canais B, e um canal mais lento (16kbps) para a transferência de dados de controlo, chamado canal D. A interface PRI tinha como principal destinatário os clientes empresariais e seguia a norma ITU-T I.431. Comparando com o BRI, o PRI oferecia mais canais B que dependiam dos padrões locais - na Europa era oferecida uma taxa de 2048kbps que correspondia a 30 canais B mais um canal D.

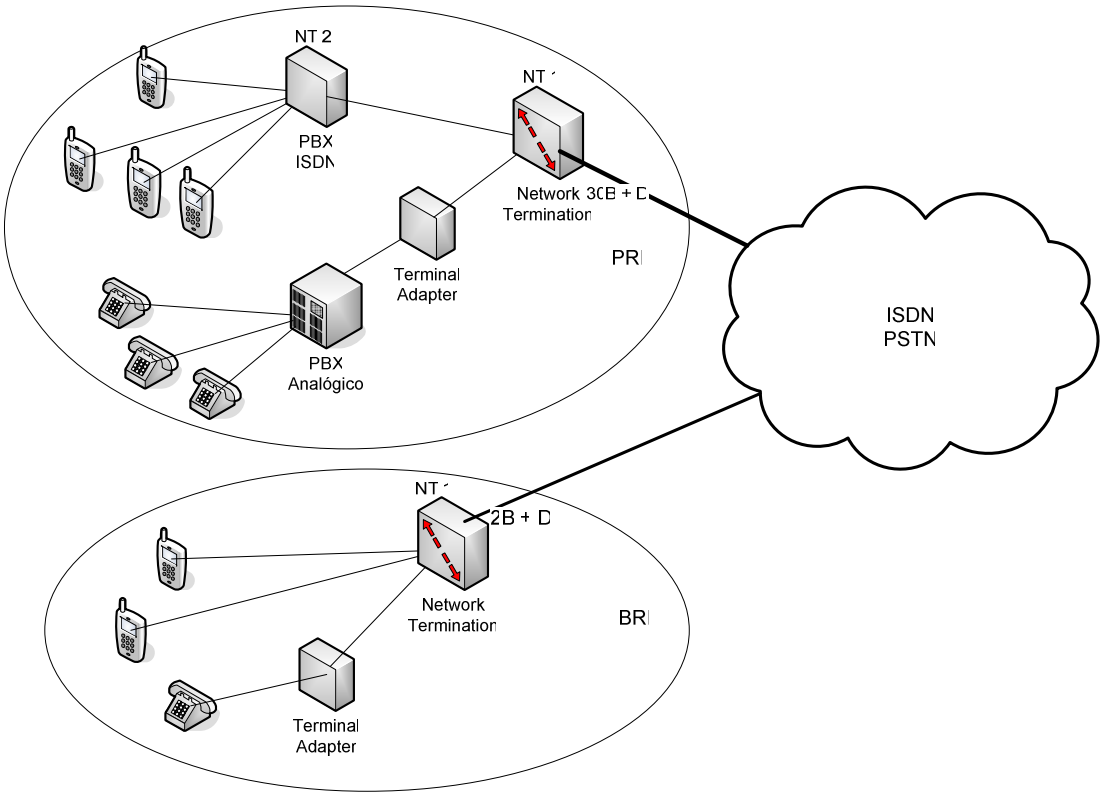


Figura 4 – Acesso RDIS

Para rentabilizar a infra-estrutura existente (rede de cobre) foi desenvolvida outra tecnologia digital que permite a transferência de dados com débitos mais elevados do que a RDIS. Esta tecnologia é denominada xDSL onde o x varia com a técnica utilizada.

Actualmente o ADSL (*Asymetric* DSL) permite débitos descendentes de 24Mbps e ascendentes de 2Mbps[1], embora estes valores dependam do comprimento e da qualidade da linha. A tecnologia DSL usa a parte não utilizada do espectro disponível no lacete local. O espectro disponível é dividido em canais de 4312,5 Hz e cada canal é avaliado quanto à sua disponibilidade e usabilidade. Quanto maior for o número de canais, maior será a largura de banda disponível. Os canais são classificados de *upstream* ou *downstream*, ficando o de *upstream* na faixa entre os 25,875kHz e os 138kHz e o *downstream* na faixa entre os 138kHz e os 1104kHz.

O equipamento do cliente transforma os sinais analógicos e digitais num sinal DSL e transmite-os através do modem DSL. O lacete local transporta o sinal até à estação onde o sinal é dividido, a parte analógica (voz) é enviada pela rede POTS e a parte digital (dados) vai para o *Digital Subscriber Line Access Multiplexer* (DSLAM) onde são agregados a outros sinais e enviados para o respectivo ISP.

Existem vários tipos de DSL: ADSL, HDSL e VDSL.

O ADSL é definido pela norma ITU-T G. 992.1 e ITU-T G.992.2. O ADSL 2 é uma evolução do ADSL, permitindo um maior débito, alcance e diagnósticos, sendo definido pelas normas ITU-T G. 992.3 e ITU-T G.992.4. Uma nova variante o ADSL2+, definida pela norma ITU-T G. 992.5, duplica a largura de banda descendente, fornecendo débitos de 20Mbps até cerca da 1,5km.

O HDSL (*High bit rate Digital Subscriber Line*), definido pelo ITU-T G. 991.1 usa a *Pulse Amplitude Modulation* para a codificação e tem taxas de transferência simétricas na gama de 784 – 2320kbps. O SHDSL (*Single-pair High-Speed Digital Subscriber Line*), uma evolução do HDSL fornece débitos simétricos entre os 192kbps e os 4.6Mbps. sendo definido pela norma ITU-T G.991.2.

[1] Os valores apresentados para os débitos das tecnologias ADSL e VDSL são meramente para referência, visto que se encontram em constante evolução

O VDSL (*Very high bit rate DSL*) é definido pela norma ITU-T G.993.1 e permite débitos simétricos de 50Mbps.

2.2.2. REDE DE TRANSPORTE

A rede de transporte interliga os equipamentos de comutação de pacotes ou circuitos, usando para a transmissão de informação os meios físico, como o cobre ou a fibra óptica, ou o meio livre, como por exemplo o rádio.

As redes de transporte começaram por ser analógicas e como técnica de multiplexagem para agregar os diversos canais de voz era usada a *Frequency Division Multiplex* (FDM), ao qual cada canal de voz era modulado numa portadora.

Usando a técnica de multiplexagem *Time Division Multiplex* (TDM), em que o sinal é amostrado em intervalos de tempo regulares, convertido em código binário e agrupado no meio e transmissão, no destino é efectuada a operação inversa.

Os primeiros sistemas TDM usavam técnicas *Pulse Code Modulation* (PCM), nas quais os sinais de voz eram amostrados a 8kHz e codificados em 8 bits o que resultava num sinal codificado de 64kbps. Os *Multiplexers* resultantes agregavam 30 canais de voz mais dois de controlo a 2,048Mbps, chamado E1 (nos EUA são agrupados 24 canais e o sistema chama-se T1). Estes sistemas são conhecidos como sistemas primários e foram a base para o *Plesyochronous Digital Hierarchy* (PDH).

No PDH são formados vários níveis de transporte, no nível 1 (E1) são enviados 30 canais (2048kbps), no nível 2 (E2) que é composto por 4 E1 são enviados 120 canais (8448kbps) [Tabela 1].

Tabela 1 - Níveis PDH

Ordem	Nível de Hierarquia	Bitrate kbit/s	Nº de canais	Meio de Transmissão
1	E1	2.048	30	Cobre
2	E2	8.448	120	Cobre
3	E3	34.368	480	Fibra/Rádio
4	E4	139.264	1920	Fibra/Rádio

O sinal gerado é formado pela junção de 4 tributários de nível imediatamente abaixo, usando o entrelaçamento de bits.

O principal problema dos sistemas PDH é que, a partir do segundo nível e devido ao entrelaçamento de bits, se torna mais difícil a inserção/derivação dos tributários para ligações intermediárias da rede, sendo necessário desmultiplexar o feixe agregado, tornando o processo pouco flexível e demasiado caro.

Tendo em conta as deficiências do PDH, na década de 1980 o ITU-T padronizou o *Synchronous Digital Hierarchy* (SDH), um sistema mais flexível e com maior capacidade de gestão.

O SDH fornece às redes de transmissão altas taxas de transferência e a possibilidade de uma gestão eficiente e centralizada.

A multiplexagem, desempenho e fiabilidade das redes SDH estão totalmente referenciadas nas normas ITU-T, o que torna possível a interoperabilidade e implementação de ambientes multi-fornecedor.

Os sinais tributários PDH e outros serviços podem ser transportados pelo SDH. O SDH pode ser usado em redes de longa distancia (WAN), metropolitanas (MAN) e redes de acesso, tornando possível uma infra-estrutura de rede de transporte estruturada e unificada.

Actualmente, as redes de transporte são constituídas por redes PDH e SDH e sistemas ópticos (WDM). Estas estruturas integradas resultam numa infra-estrutura simples com interfaces padronizadas, com tempos de comutação de protecção rápido e funcionamento estável e automático. A tecnologia mais comum é em anel auto-regenerativo, que permite obter as características apontadas.

SDH-NG

Tradicionalmente o SDH é ineficiente para o transporte de dados, devido à natureza estatística desse mesmo tráfego.

A ineficiência do transporte de tráfego *Ethernet* pode ser visto na Tabela 2

Tabela 2 - Ineficiência do SDH para tráfego Ethernet

Ethernet	Sinal	Bitrate	Ineficiência
10 Base T (10Mbit/s)	STM-0	51,840 Mbit/s	81%
100 Base T (100Mbit/s)	STM-1	155,520 Mbit/s	36%
Gigabit Ethernet (1Gb/s)	STM-16	2,5 Gbit/s	60%

Para optimizar o transporte de *Ethernet* sobre SDH foram definidos elementos que aumentam a flexibilidade do SDH, de modo a suportar interfaces de dados e funcionalidades de agregação de dados. Estes elementos incluem:

- *Virtual Concatenation* (VCAT) [6] é a técnica de concatenação virtual para a agregação de canais de forma compor um canal de maior velocidade.
- *Link Capacity Adjustment Scheme* (LCAS) [7] é um método de aprovisionamento e configuração dinâmico de canais TDM de modo a adequar as necessidades de banda do utilizador.

- *Generic Framing Procedure* (GFP) [8] é um protocolo de adaptação de qualquer tipo de tráfego de dados (*Ethernet*, IP, etc.), para ser mapeado em SDH.

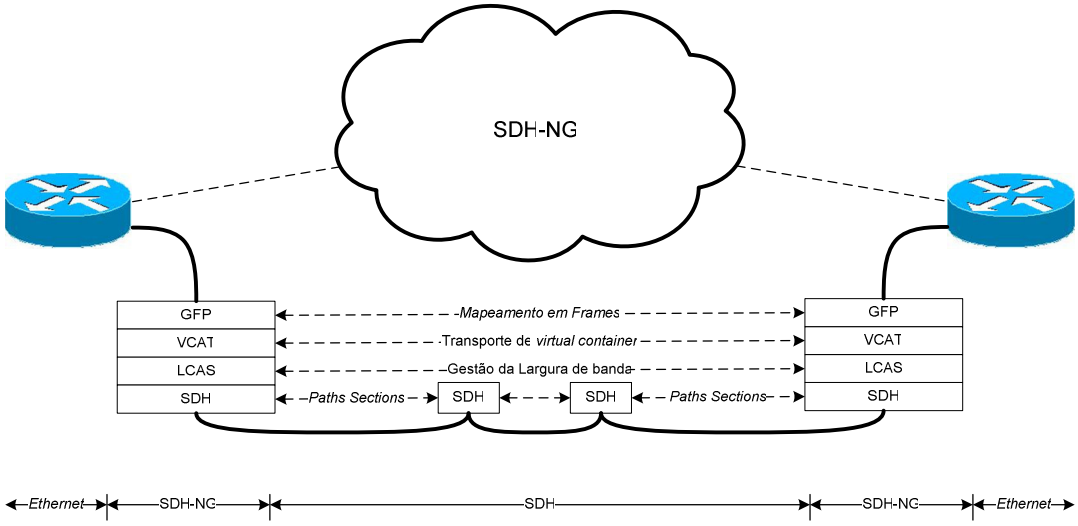


Figura 5 – SDH-NG

Estas técnicas permitem uma utilização mais eficiente dos recursos da rede SDH. A Tabela 3 mostra o ganho em eficiência no transporte de dados com ênfase no VCAT. Por exemplo para passar tráfego *Ethernet* de 10Mbit/s aloca-se 5xVC-12 (2 Mbps) concatenados virtualmente, obtendo assim uma taxa de utilização de 100%.

Tabela 3 - Eficiência do VCAT

Protocolo do Cliente	Bitrate (Mbps)	SDH	Utilização	C/ VCAT	Utilização
Ethernet	10	VC-3	20%	VC-12-5v	100%
Fast Ethernet	100	VC-4	67%	VC-3-2v	99%
Gigabit Ethernet	1000	VC-4-16c	42%	VC-4-7v	95%

Tal como o IP se tornou a interface dominante ao nível dos serviços, existe uma tendência para que isto se reflicta com a Ethernet no nível físico. Muitos

serviços estão a ser criados sobre *Ethernet*, fazendo com que haja uma corrente para que o *Ethernet* se torne universal para a rede de transporte, de modo a evitar conversões de formatos, que se traduzirá numa redução de custos.

No futuro "*Ethernet* Nativa" e "*Ethernet* sobre SDH" coexistirão sobre a rede de transporte, tendo papéis complementares. Nas redes de acesso, iniciativas como *Ethernet in the First Mile* (EFM) irão permitir um transporte eficiente. Ao mesmo tempo, nas situações onde terão de ser usados circuitos TDM ou necessidade de acesso protegido, a solução *Ethernet* sobre SDH é mais eficiente. No core da rede onde a protecção, operação e manutenção são críticos, o modo predominante de transporte será o SDH/EoSDH [9].

2.3. REDES DE PRÓXIMA GERAÇÃO

As Redes de Próxima Geração (NGN – *Next Generation Networks*) estão a ser desenvolvidas por diversas agências e organizações um pouco por todo o mundo. A maior parte delas está a trabalhar em conjunto na criação de um sistema uniformizado e *standard*.

Existem visões diferentes acerca dos sistemas de comunicação de próxima geração. A maior parte delas tentam fornecer ao utilizador serviços e funcionalidades similares, mas optando por aproximações diferentes.

De seguida serão analisados os requisitos para os sistemas NGN definidos pelo ITU-T e apresentadas as diferentes organizações que estão a trabalhar no sentido de obter uma solução *standard*. Das várias agências que estão a desenvolver as redes de próxima geração, a maior contribuição vem do 3GPP e do 3GPP2, que apresentaram o IMS – (IP *Multimedia Subsystems*), que é a base para o trabalho realizado por outras organizações.

Não existe uma definição precisa para as NGN, sendo talvez a melhor definição a que se encontra na norma ITU-T Y.2001 [10]:

A Rede de Próxima Geração (NGN) é uma rede de pacotes capaz de fornecer serviços de telecomunicações fornecidos em banda larga. As tecnologias de transporte terão de suportar QoS, nas quais as funções relacionadas com serviços terão de ser independentes das tecnologias de transporte usadas. Permitir acesso livre de utilizadores a redes e a fornecedores de serviço concorrentes. Suporta mobilidade generalizada, que irá permitir um fornecimento de serviços consistente e ubíquo.

As NGN podem ser definidas através das seguintes características:

- Transporte baseado em comutação de pacotes;
- Separação das funções de controlo nas funcionalidades de suporte, chamada/sessão e aplicação/serviço.
- Desacoplamento do aprovisionamento de serviços do transporte e aprovisionamento de interfaces abertas;
- Suporte a uma ampla gama de serviços, aplicações e mecanismos baseados em criação de serviços modulares (incluindo *real time/streaming/non-real time* e serviços multimédia);
- Funcionalidades de banda larga com QoS *end-to-end*
- Interligações com rede *legacy* através de interfaces abertas;
- Mobilidade generalizada;
- Acesso livre a diferentes prestadores de serviço;
- Esquemas de identificação variados;

- Características de serviços unificadas, para serviços que aparentem ser iguais para o utilizador;
- Convergência de serviços fixo/móvel;
- Independência de funções relacionadas com o serviço e a tecnologia que suporta esses serviços;
- Suporte a múltiplas tecnologias para a ligação ao cliente;
- Compatível com todos os requisitos das entidades reguladoras, como por exemplo as comunicações de emergência, segurança, privacidade, etc.

2.4. MODELO FUNCIONAL ITU-T G.805

As redes de transporte actuais são compostas por múltiplas tecnologias, tais como ATM, o SDH e o OTN (*Optical Transport Network*), que operam em várias camadas e numa relação cliente/servidos. A recomendação ITU-T G.805 [11] define um modelo funcional para a rede de transporte de uma forma genérica e independente da tecnologia.

O modelo de uma rede de transporte é baseado no conceito de separação por camadas e compartimentada, onde as camadas interagem numa relação cliente/servidor. A norma ITU-T G.805 define um modelo genérico por camadas e compartimentada.

Os componentes que compõem o modelo ITU-T G.805 estão apresentados na Figura 6. Quatro entidades de transporte fornecem o transporte entre os pontos de referência:

- A *link connection* transporta a informação transparentemente através do *link*. É composta por um par de pontos *Adaptation* e um *Trail* na camada servidor;

- A *subnetwork connection* transporta a informação transparentemente através de uma *subnetwork*. A *subnetwork connection* é uma concatenação de *subnetwork connections* e de *link connections*;

- A *network connection* transporta a informação transparentemente através da *layer network*. É formada pela concatenação de *link connections* e/ou *subnetwork connections* entre *connection points* terminais;

- O *Trail* transporta informação adaptada e monitorizada da camada cliente entre os *access points*;

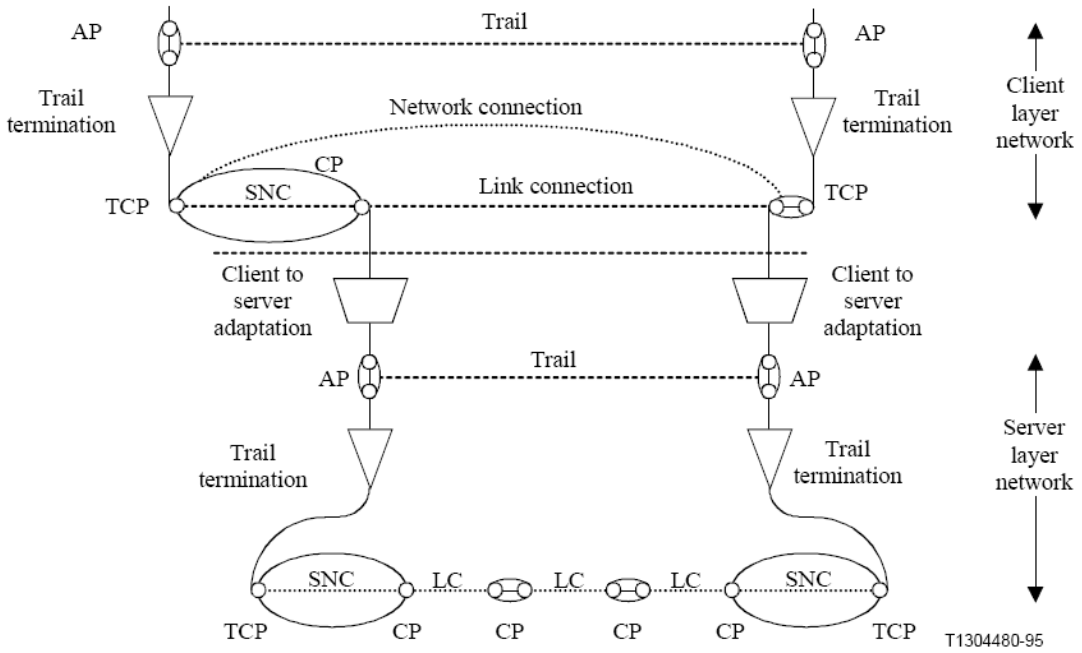


Figura 6 – Modelo Funcional ITU-T G.805 [11]

Em cada camada são necessárias funções de transporte para descrever a rede de transmissão.

- A *adaptation function* representa o processo de conversão entre a camada servidor e a camada cliente. A adaptação do sinal inclui, codificação e *framing*. A *adaptation function* também é usada para a *multiplexagem, desmultiplexagem* e *multiplexagem* inversa;

- A *termination function* realiza a supervisão da integridade do sinal da camada. Esta operação é realizada adicionando informação de monitorização na *source*, tal como o *checksum* e analisando esta informação na *termination function sink*.

O ITU-T G.805 define dois tipos de *layer network*: o *path layer network* e a *transmission* media *layer network*. O *path layer network* fornece as funcionalidades de transmissão para dar suporte a diversos tipos de serviços do cliente independentes da *transmission* media *layer network*. A *transmission media layer network* é suportada por *trails* e *link connections* e podem depender do meio físico usado para a transmissão (fibra óptica ou rádio). A Figura 7 mostra o *path layer* (cinza) e a *transmission* media *layer*.

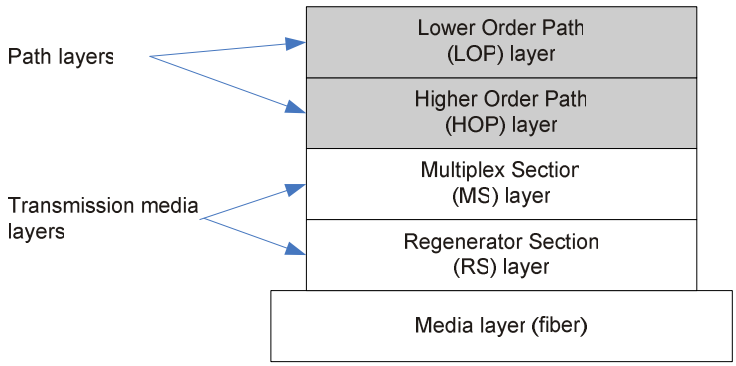
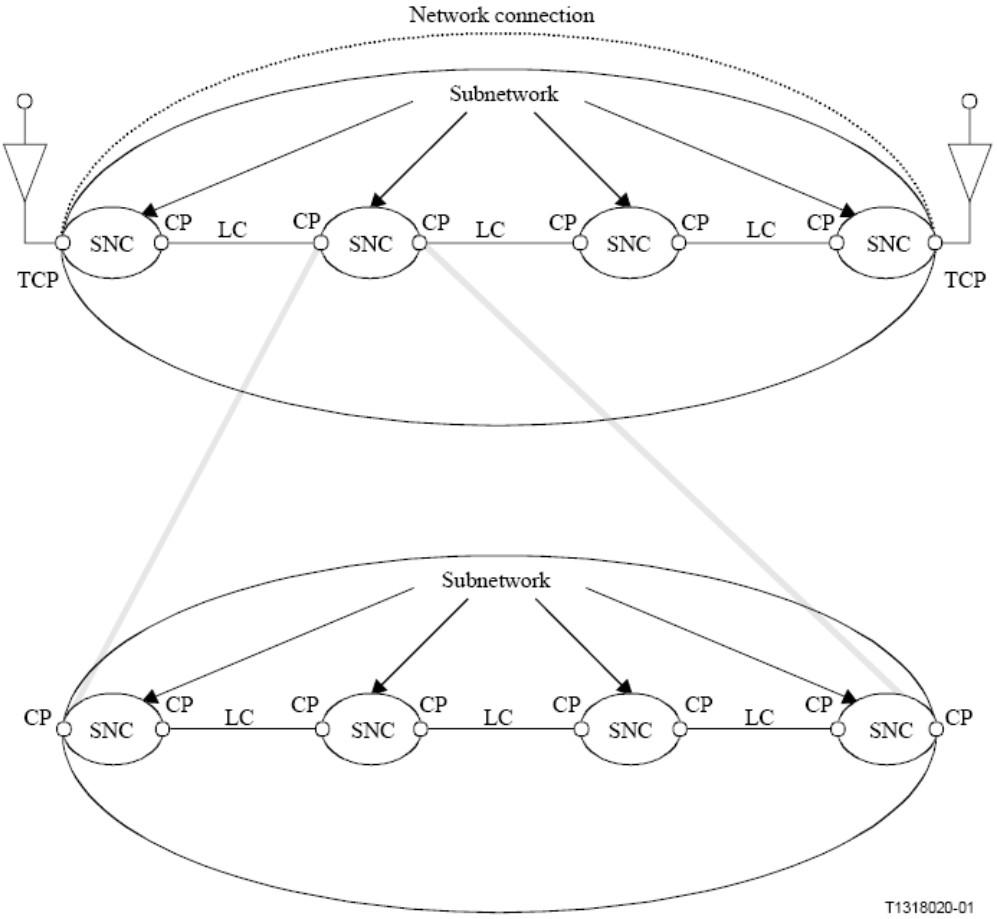


Figura 7 – *Network layer* SDH

Juntamente com os conceitos de separação por camadas (*layering*) a referência ITU-T G.805 define o conceito de compartimentação (*partitioning*) [Figura 8] para representar a estrutura organizacional dentro de uma camada. O

conceito de compartimentação é baseado no conceito de decomposição recursiva da *network layer* em *subnetworks* e de *subnetworks* em *subnetworks* mais pequenas e *link connections*. A compartimentação da rede pode ter em conta estruturas administrativas ou estruturas organizacionais.



CP Connection Point
LC Link Connection
SNC Subnetwork Connection
TCP Terminator Connection Point

Figura 8 – Conceito de Partitioning [11]

2.5. MODELOS DE OPERAÇÕES E SERVIÇOS

2.5.1. OSS

Os Sistemas de Suporte às Operações (OSS – *Operations Support Systems*) são entidades lógicas que representam o sistema de gestão da rede de telecomunicações [12]. Engloba um conjunto de processos que um Telco (Operador de Telecomunicações) necessita para o aprovisionamento, monitoria, controlo e análise da rede, para gerir o controlo de falhas e realizar funções que envolvam a interacção com os clientes.

Para tratar dos processos mais ligados aos clientes foi definido o termo *Business Support Systems* (BSS), no qual estão englobados todos os processos que um Telco necessita para o relacionamento com agentes externos – clientes, parceiros e fornecedores. A fronteira entre o OSS e o BSS é praticamente indistinta, podendo as funções BSS serem consideradas a parte do OSS orientada ao cliente. Por isso, os sistemas de suporte à operação são referidos como OSS/BSS ou simplesmente pela sigla OSS, que engloba os dois termos.

Os sistemas OSS são uma parte complexa e crítica das funções de um Telco. Tem havido um esforço na padronização dos sistemas OSS, tendo várias entidades chegado a um grau de uniformização satisfatório.

Os sistemas de gestão podem variar entre simples formulários, gerindo um número limitado de elementos e sistemas complexos, compostos por elementos de rede e elementos de serviço [Figura 9].

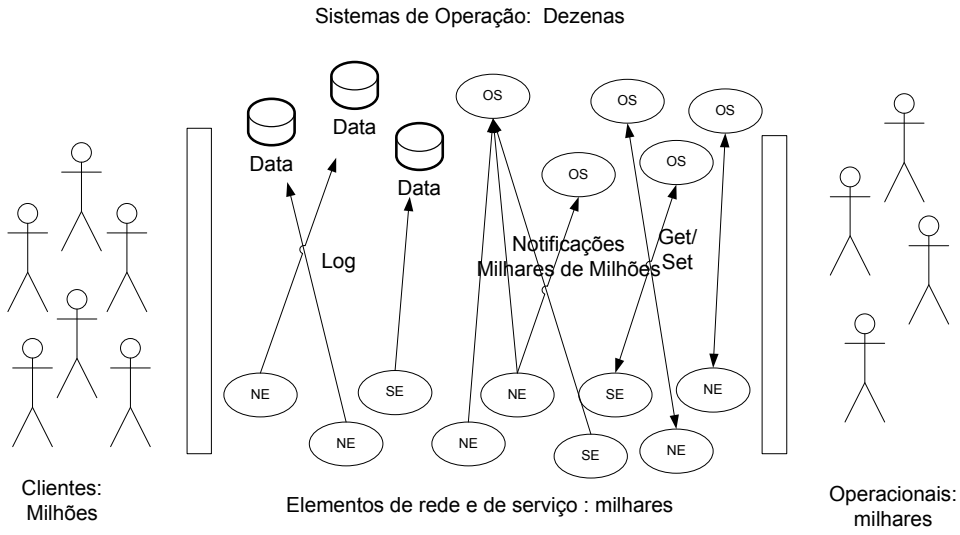


Figura 9 – Elementos de um sistema de gestão

Um elemento de rede (NE – *Network Element*) é um equipamento gerido. O software incluído é parte de uma infra-estrutura de rede. Entre os exemplos de elementos de rede encontram-se os comutadores, concentradores e os sistemas de transmissão SDH.

Um elemento de serviço (SE – *Service Element*) é o software ou elemento de rede responsável por fornecer um dado serviço, por exemplo os pontos de controlo (SCP) de uma rede inteligente (IN).

As funções de suporte às operações estão divididas em gestão de falhas, aprovisionamento e facturação. Cada uma destas funções é realizada por software de suporte às operações (OS – *Operations System*). Estes OS requerem grandes transferências de dados, entre os NE e os SE, que são suportados por bases de dados.

Os sistemas de gestão interagem com todos os elementos, incluindo os clientes e os operacionais. Como os sistemas que geram são complexos e as operações realizadas são em grande escala, isto faz com que os sistemas de gestão sejam bastante complexos. Devido à sua complexidade os sistemas de gestão estão

dispostos em camadas como se pode ver na Figura 10 onde são apresentadas as camadas do *Telecommunications Management Network* (TMN). [13]

Figura 10 – Divisão em Camadas segundo o modelo TMN [13]

As normas para os sistemas de gestão tentam responder às seguintes questões:

- Quais os elementos, sistemas ou serviços a gerir?
- Qual a informação que é necessária para gerir esses elementos, sistemas ou serviços. Como é que essa informação poderá ser definida de modo a permitir sistemas de gestão estruturados e abertos?
- Como é que um elemento gerido interage com um elemento gestor? Que protocolos e APIs é que são necessários?
- Que protocolos de comunicação podem ser usados com os sistemas de gestão?
- Que funcionalidades de gestão são necessárias?

Os primeiros sistemas de gestão abordavam estes problemas de uma perspectiva *bottom-up*, focando os protocolos e a informação. Os novos padrões passaram a abordar o problema numa perspectiva *top-down*.

Evolução dos sistemas OSS

Os objectivos e a natureza dos sistemas de gestão foram mudando ao longo do tempo.

O modelo de gestão IETF usava um sistema simples que consiste num agente gestor e numa comunicação, baseada em protocolos, entre o gestor e a Entidade Gerida [Figura 11]. O elemento a ser gerido é representado por um conjunto pré definido de parâmetros, como parte de uma estrutura de informação chamada de *Management Information Base* (MIB).

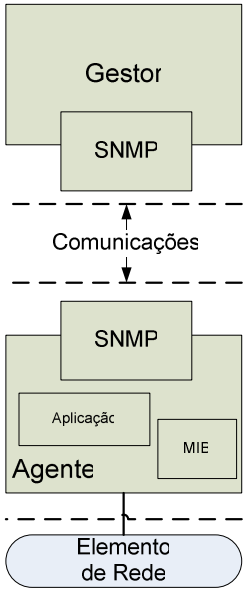


Figura 11 – Modelo *Manager – Agent*

Seguidamente, os operadores quiseram gerir subsistemas dentro de uma rede, como por exemplo os sistemas de transmissão SDH, um conjunto de *switches*

TDM ou uma rede SS7. Os sistemas de gestão passaram a estar focados nos elementos e em como eles interagiam com o sistema [Figura 12]. Com uma perspectiva ao nível da rede, os serviços prestados passaram a necessitar de ser geridos. Os sistemas de gestão requerem a separação em camadas. A camada de rede, por exemplo, que coordena e controla os elementos e a camada de serviço que gere os diferentes serviços. Para aprovisionar um circuito alugado, será necessário alocar os recursos SDH que suportem os requisitos (largura de banda) desejados.

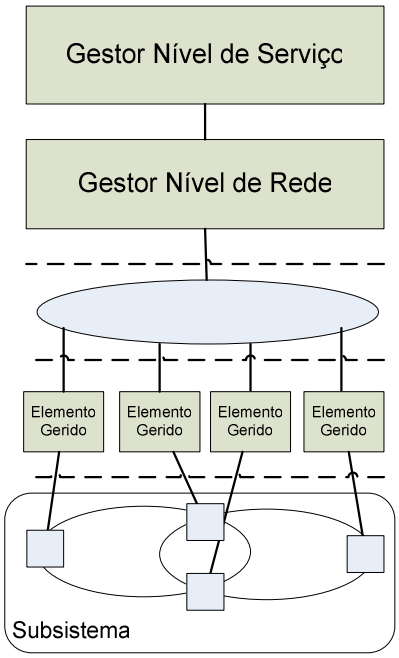


Figura 12 – Modelo com Subsistemas

Actualmente, para suportar uma gestão integrada, o sistema de informação é partilhado por várias aplicações de gestão [Figura 13]. Os sistemas de gestão estão estruturados de forma a que haja uma separação entre funções genéricas e os processos de negócio. A funcionalidade passou a ser modular e os sistemas passaram a ser distribuídos.

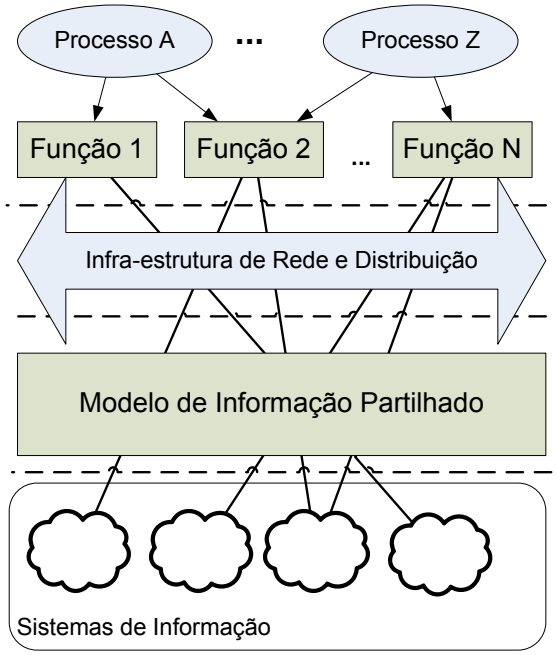


Figura 13 – Modelo Distribuído

Em seguida vão-se abordar genericamente os vários padrões que foram aparecendo, mais precisamente o *Telecommunication Management Network* e as iniciativas da TMForum (TOM e actualmente o NGOSS).

Telecommunication Management Network (TMN)

Proposto pelo ITU-T este modelo foi desenvolvido como parte de um conjunto de normas de gestão de redes [13]. Foi desenvolvido em camadas, sendo a forma mais comum de apresentação em pirâmide [Figura 14]. Cada camada da pirâmide depende somente da camada inferior e oferece funcionalidades e serviços à camada superior.

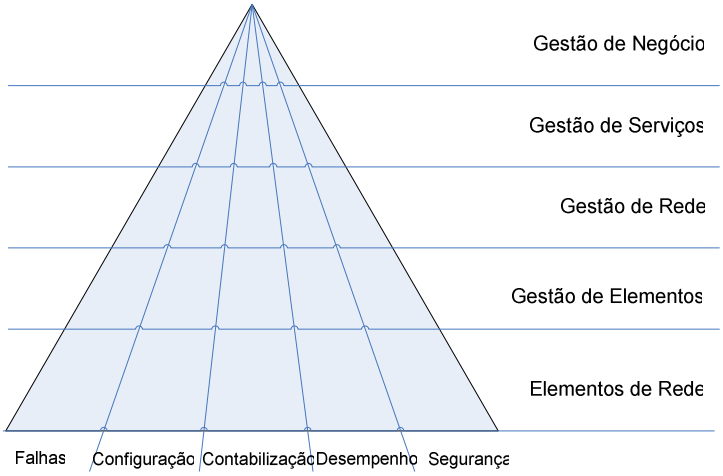


Figura 14 – Pirâmide TMN

Este modelo é uma tentativa de dividir um serviço em várias camadas estanques, que interagem com outros serviços de forma independente, mas mantendo uma interface consistente entre camadas.

As camadas definidas pela norma são:

1) Elementos de Rede (*Network Elements*): rede física do operador, composta pelos equipamentos (*switches*, *multiplexers*, …) que são referidos pela norma como recursos;

2) Gestão de Elementos (*Elements Management*): conjunto de funcionalidades necessárias para gerir um elemento. Composto por soluções proprietárias a cada elemento e soluções abertas que permitem gerir os equipamentos individualmente (soluções baseadas no protocolo SNMP, por exemplo). Os processos suportados nesta camada são de configuração básica de cada equipamento (activação e desactivação de portos, configuração de rotas e circuitos virtuais, etc.);

3) Gestão de rede (*Network Management*): controla o conjunto de elementos que formam uma ligação *end-to-end*. As soluções usadas são uma extensão das utilizadas na gestão de elementos, oferecendo uma visão mais completa da rede. Nesta camada também se encontra o inventário de redes. Os processos presentes nesta camada são a medição da performance da rede, a configuração *end-to-end*, de rotas e circuitos virtuais, optimização de *routing*, etc.

4) Gestão de serviços (*Service Management*): camada onde se encontram as actividades necessárias ao controlo dos serviços oferecidos. As ferramentas usadas estão ligadas à camada imediatamente superior. Os processos suportados por esta camada incluem o controlo da QoS, correlação entre os serviços e a rede, etc.

5) Gestão de negócio (*Business Management*): composto por todas as actividades de gestão de clientes, os processos usados são implementados em sistemas CRM, por exemplo, cobrança, *billing*, mediação etc.

Conjuntamente foram importadas do modelo OSI as cinco áreas funcionais de gestão, conhecidas pelo acrónimo FCAPS (*Fault*, *Configuration* , *Account*, *Performance* e *Security*) que serão descritos de seguida:

1) Gestão de Falhas (*Fault Management*): detecta, isola e corrige condições anormais na rede e nos elementos de rede. Cobre operações como, vigilância de alarmes, detecção de falhas, isolamento e controlo, *logging*, etc;

2) Gestão de Configurações (*Configuration Management*): fornece os dados e controla os elementos de rede de modo a obter o comportamento pretendido. Inclui as funções gestão de vistas, gestão

de topologia, gestão de software, gestão de inventário, aprovisionamento, estado e controlo;

3) Gestão de Contabilização (*Accounting Management*): mede a utilização dos serviços e determinação de custos, incluindo a utilização de recursos, colecta de dados de cobrança;

4) Gestão de Desempenho (*Performance Management*): avalia e realiza relatórios dos comportamentos e eficiência da rede, equipamentos ou elementos de rede. Como funções inclui a avaliação da eficiência dos recursos, gestão de tráfego e monitorização do desempenho;

5) Gestão de Segurança (*Security Management*): controla o acesso dos clientes a dados e recursos, detecção e *tracking*, reporta as violações de segurança e manutenção de serviços de segurança, por exemplo encriptação;

O TMN serviu de guia para a criação de sistemas de gestão de redes de telecomunicações. A organização FCAPS para as funções de gestão tem sido amplamente adoptada. A ênfase no TMN recai sobretudo nas camadas de elemento e de rede, estando a gestão de serviços menos desenvolvida.

A identificação de funções que suportem os processos da camada de negócio é útil, mas não se encontra explorado no TMN. A conceptualização de dessa área começou a ser feita pelo TOM (*Telecommunication Operations Map*).

Telecommunication Operations Map

O TOM é uma resposta às limitações do TMN, mas também é o percursor do sistema eTOM (*enhanced* TOM). O principal objectivo do TOM foi transformar a visão em camadas do sistema TMN num modelo de processos.

Ao desenvolver o TOM e o eTOM o TMForum focou apenas os processos de negócio que um operador necessita, a funcionalidade e a informação necessária para suportar esses processos e como o sistema de gestão pode ser integrado usando componentes *third party*. O modelo tenta dar resposta às necessidades de negócio de todos os Telcos.

O TOM identifica as funções que são necessárias para três grupos de processos que um operador precisa para a sua relação com os clientes [14]. O operador ao fazer um pedido para um serviço tem de realizar as seguintes tarefas: fazer um aprovisionamento dos recursos necessários, iniciação da cobrança e activação do serviço. Uma vez activo, tem de ser assegurada a sua operacionalidade e QoS, dando resposta a falhas e degradação do serviço. Durante a utilização do serviço tem de ser recolhida informação sobre a utilização. Ao cliente terão de ser cobradas as tarifas acordadas. Os processos nas três áreas *fulfillment*, *assurance* e *billing* (FAB) requerem um número de funções genéricas. A tarefa do TOM é identificar estas funções, agrupá-las em processos e mapear as interacções entre processos de modo a criar processos de gestão *end-to-end*.

As funções necessárias para um processo FAB envolvem interacções com o cliente, serviços, rede de recursos e ao nível do elemento de rede. Funções de diferentes camadas têm de interagir de modo a executar um processo. As camadas TMN são adoptadas pelo TOM como se pode ver na Figura 15, havendo uma alteração no nome de modo a reflectir melhor a função subjacente. As áreas FAB atravessam as camadas, agrupando as funções da seguinte forma:

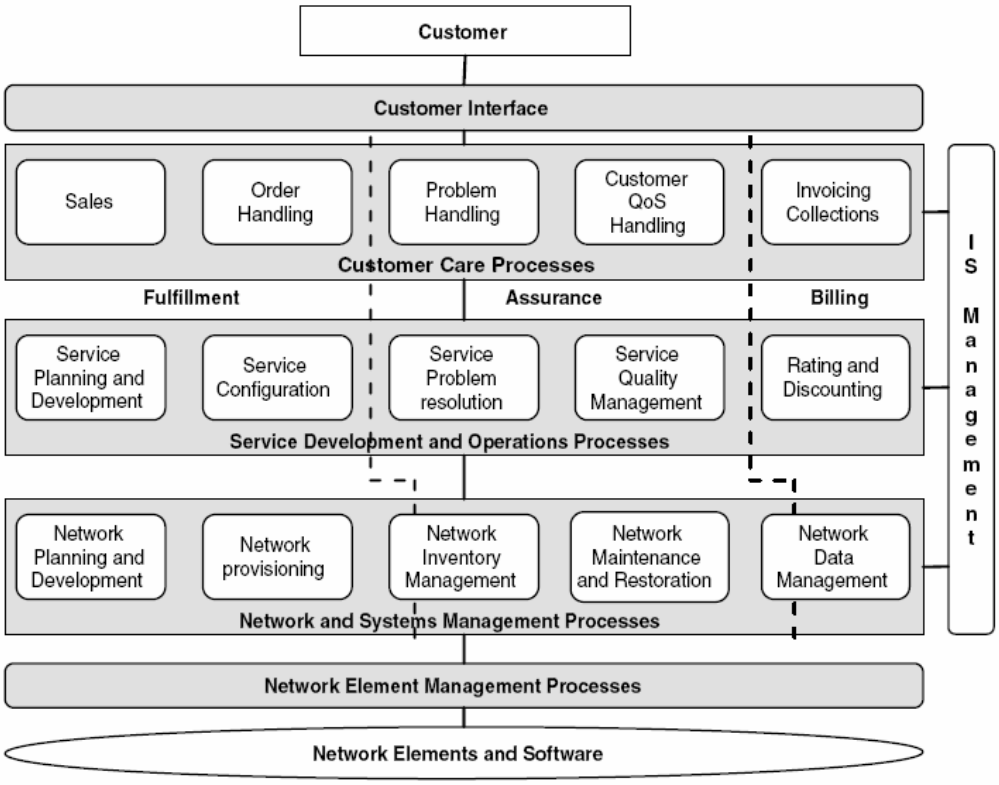


Figura 15 – ***Telecommunication Operations Map*** **[14]**

1) Processos de Satisfação do Cliente (*Customer Care Process*): é obtida da divisão da camada *service management* do TMN. As funções que tratam da compra de um serviço e handling do pedido estão na área de *fulfillment*, o *problem handling* e QoS estão na *assurance*, o *invoicing* e *collection* estão na área de *billing*. Todas estas funções envolvem interacções com o cliente através de um processo de interface com o cliente.

2) Processos de Desenvolvimento e Operação de Serviços (*Service Development and Operation Process*): é um desenvolvimento da camada *Service Management* do TMN. Tal como a anterior os grupos FAB estão distribuídos pelas funções *Service Planning and Development*, *Service Configuration*, *Service Problem Resolution*, *Service Quality Management* e *Rating and Discounting*;

3) Processos de Gestão de Redes e Sistemas (*Network and Systems Management Process*): é um desenvolvimento da camada *Network Management* do TOM com os grupos FAB distribuídos pelas funções *Network Planning and Development, Network Provisioning, Network Inventory Management, Maintenance and Restoration* e *Network data Management*;

4) Processos de Gestão de Elementos de Rede (*Network Element Management Process*): não possui funções, representando a camada de *Element Management* do TOM.

O TOM tenciona cumprir a promessa do TMN em que a funcionalidade de uma camada inferior é acessível a uma camada acima e satisfazendo as necessidades dos processos de negócio. Para cumprir este objectivo, cada uma das camadas tem de ter funcionalidades adequadas. Estas funcionalidades têm de ser identificadas de modo a permitir que diferentes processos sejam sintetizados usando as funções existentes. A forma como as funções interagem não devem ser forçadas por pontos de referência e protocolos muito limitativos. Em vez disso deve-se adoptar uma abordagem *top-down* do *Model Driven Architecture*.

O TOM é mais recentemente substituído pelo eTOM. Que apresenta uma visão expandida dos pontos de vista empresariais e de informação. O eTOM não existe de forma isolada, fazendo parte de uma estratégia para definir a nova geração de OSS (NGOSS).

2.5.2. NGOSS – CONCEITOS GERAIS

O programa NGOSS [15] foi desenvolvido pelo TMForum, como uma norma para a criação e desenvolvimento de componentes OSS/BSS de uma forma flexível, distribuída e integrada.

O TMForum, apontou vários objectivos para o NGOSS, entre eles, a necessidade de uma framework que suporte o desenvolvimento de soluções OSS. A transição de uma framework para um sistema funcional é obtida, especificando, adquirindo e integrando um conjunto de componentes de software, que em conjunto criam processos *end-to-end*. Estes componentes terão de ser implementados usando tecnologias de software existentes ou componentes e os métodos de integração terão de ser uma solução *cost-effective*.

As recomendações para a arquitectura do NGOSS foram divididas em 4 pilares [Figura 16]:

- Processos, com o objectivo de mapear os processos de negócio que suportem os serviços de telecomunicações;
- Aplicações, que têm o objectivo de mapear todas as aplicações necessárias para suportar os processos identificados;
- Arquitectura tecnológica e de infra-estrutura, é a iniciativa com vista a desenvolver uma visão de alto nível da arquitectura de redes e sistemas que irão implementar as aplicações mapeadas;
- Modelo de dados, é o projecto que desenvolve um modelo de dados universal.

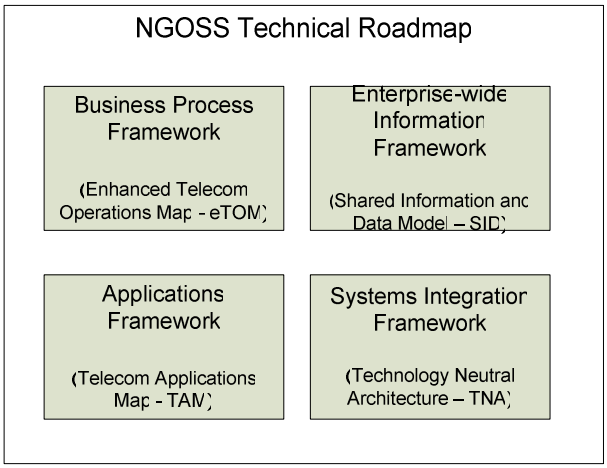


Figura 16 - Arquitectura NGOSS [16]

Enhanced TOM (eTOM)

O TOM permite a unificação dos processos, i.e., os processos que interagem com o cliente e suporte à operação, processos que interagem com a infra-estrutura, relacionados com serviços específicos. A framework permite o desenvolvimento de processos *end-to-end* de modo a satisfazer o cumprimento do serviço e os requisitos de taxação.

No entanto, os processos FAB são apenas alguns dos processos necessários pelo operador. Assim, o TOM foi expandido de modo a formar o *enhanced* TOM (eTOM) de forma a cobrir todos os processos necessários pelo operador [17].

O eTOM [18] usa uma estrutura semelhante à do TOM. O TOM tem dois níveis principais de funcionalidades, as camadas funcionais e os processos individuais. Devido à crescente complexidade do eTOM em relação ao TOM foram criados três níveis de funcionalidades. O nível zero que pode ser visto na Figura 17, fornece uma ampla classificação dos processos e identifica os *stakeholders* externos. Os processos do tipo FAB são descritos como *Operational*. Outros processos essenciais necessários aos operadores, envolvem decisões estratégicas em questões de serviços a serem oferecidos e as infra-estruturas necessárias ao suporte desses serviços. Tanto os serviços como as infra-estruturas atravessam várias fases durante o seu ciclo de vida, que necessitam de uma gestão sistemática. Para isso foi adicionada a área *Strategy*, *Infrastructure and Product* ao mapa de processos, que cobrem e suportam as decisões sobre que serviços disponibilizar, como os fornecer e como os apresentar aos clientes.

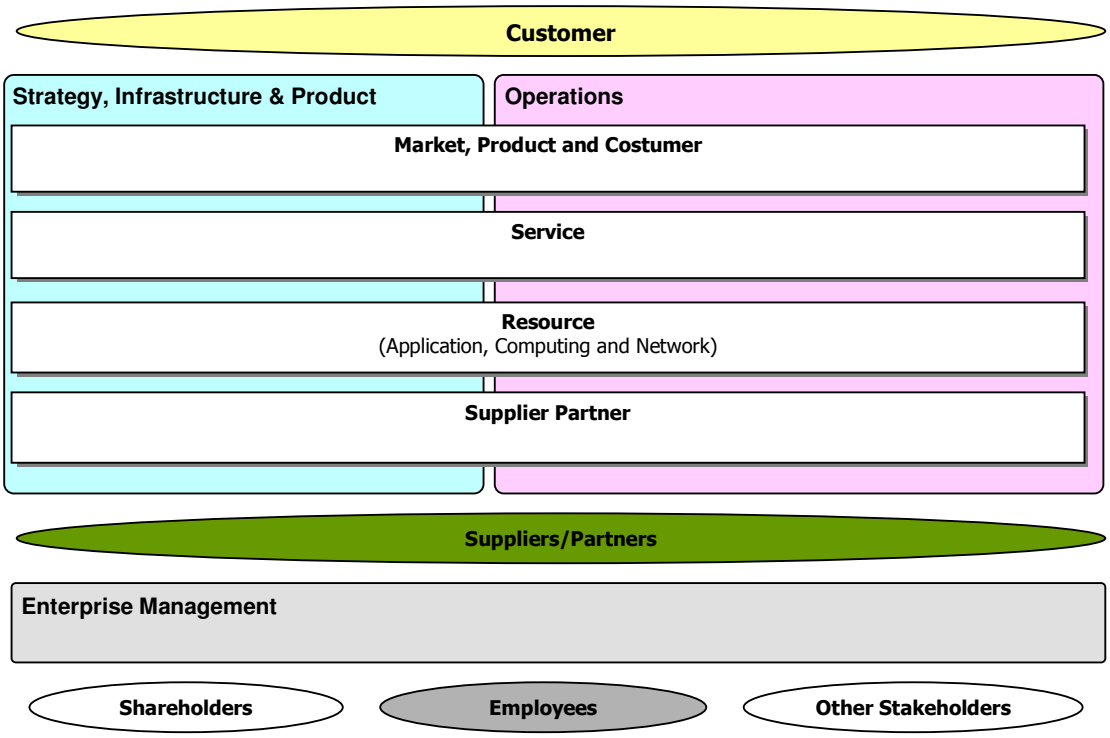


Figura 17 – Nível zero do modelo eTOM [17]

A área *Enterprise Management* contém todos os processos necessários numa organização. Tal como no TOM o mapa identifica os diferentes clientes como grupos *stakeholders*.

As funcionalidades de nível 1 (Figura 18) adicionam uma classificação vertical aos processos, agrupando-os nas áreas FAB. Na área *Operations*, são necessários processos que assegurem a resposta aos pedidos de serviço por parte dos operadores. Ao nível da relação com os clientes os canais de venda têm de ser abertos e funcionalmente eficientes. Os sistemas e os recursos necessários para fornecer um serviço terão de operar de uma forma *cost-effective*, dentro das especificações de qualidade e serem devidamente reportados. Também são necessários requisitos semelhantes ao nível de rede: os recursos têm de estar disponíveis e operacionais, de modo a endereçar novos serviços. Estes processos formam o grupo *Operational Support and Readiness* (OSR). Os processos FAB e OSR

reunem os requisitos operacionais dos operadores. As camadas horizontais são especializadas no nível 1 para as áreas operacionais e estratégicas.

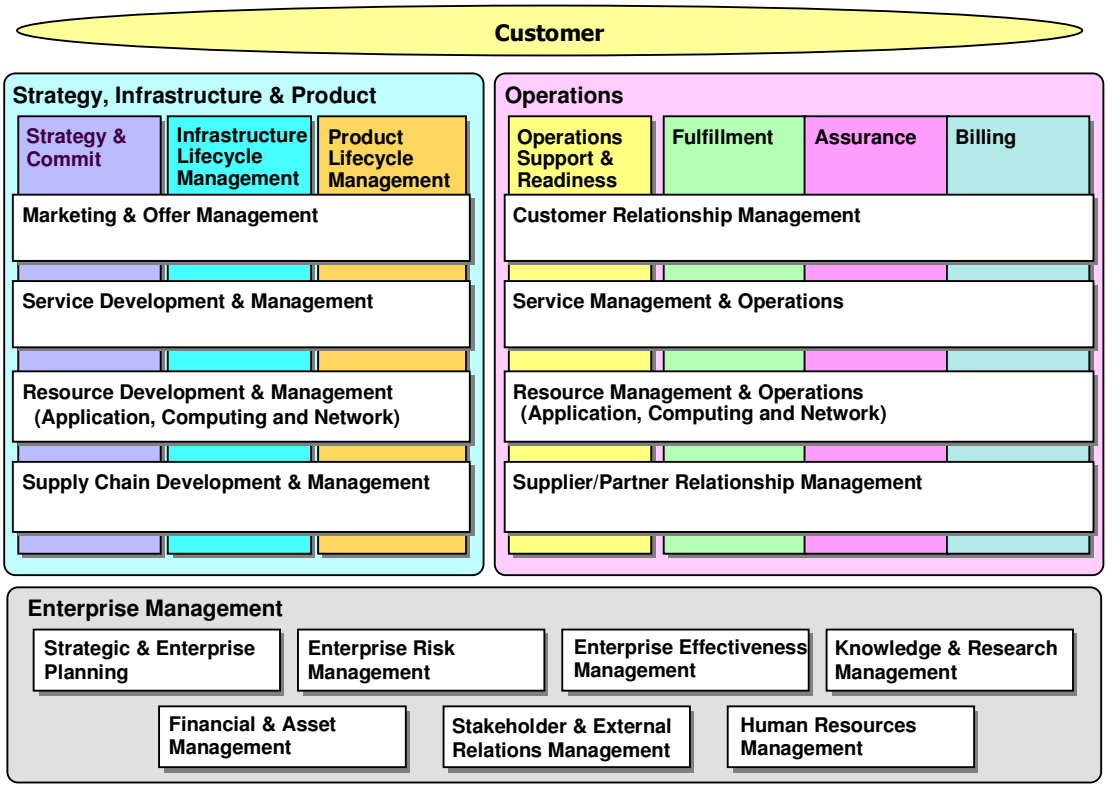


Figura 18 – Nível 1 do modelo eTOM [17]

Processos tais como *Order Handling* e *Service Configuration* formam o nível 2 do eTOM. Os processos podem ser posteriormente decompostos: os elementos dos processos de nível 2 são decompostos em elementos de nível 3, que por sua vez podem ser decompostos em elementos de nível 4.

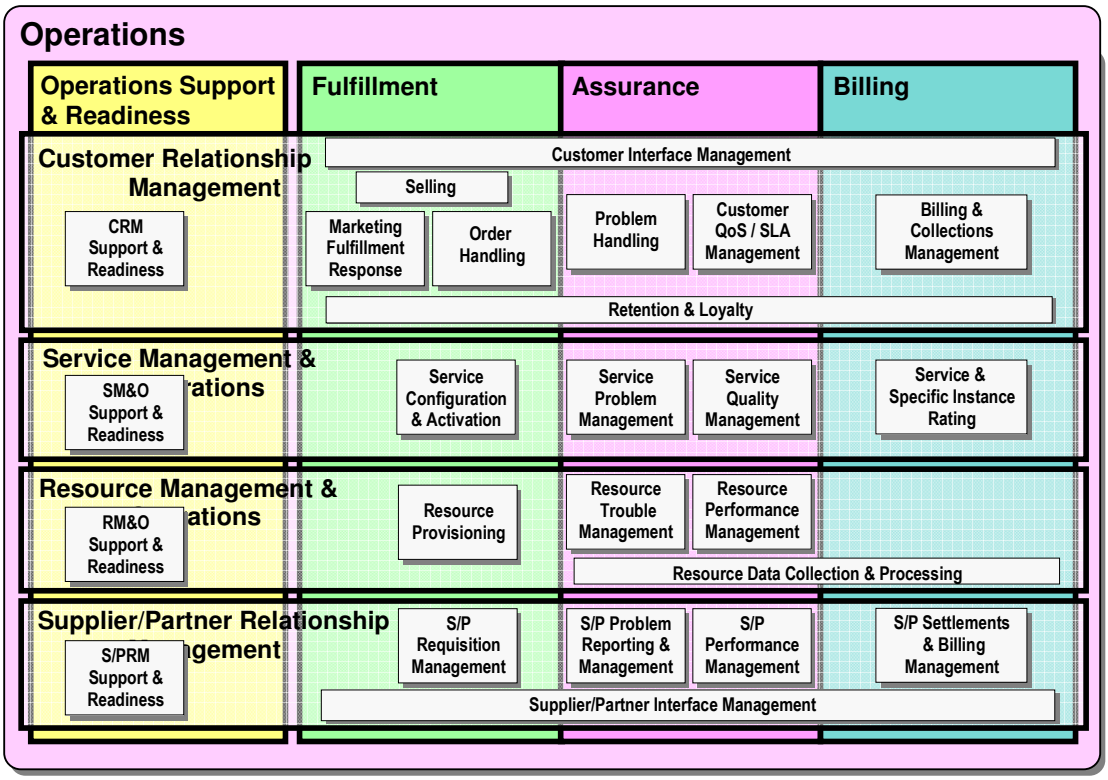


Figura 19 – Nível 2 do modelo eTOM [17][2]

Modelo SID *(Shared Information/Data)*

O modelo SID [19] antes de ser um modelo de dados é um modelo de informação. O modelo de informação SID representa a vista lógica dos objectos de interesse (entidades), tais como cliente, localização e elemento de rede. Também representa as relações (associações) entre entidades, tais como uma rede está numa dada localização. As entidades são depois caracterizadas por atributos que as descrevem e comportamento (operações) que descreve como as entidades operam. Como modelo de dados o modelo SID representa a implementação física da visão lógica dos objectos [20]. É usado para representar diferentes perspectivas da informação, por exemplo ao nível do negócio ou ao nível do sistema. O modelo SID fornece um modelo de referência para informação/dados e um vocabulário

[2] Em anexo (Figura 34) encontra-se a figura completa do nível 2 do modelo eTOM

comum de informação/dados de um ponto de vista de sistema e de negócio e usa UML para formular as necessidades dos pontos de vista requeridos. A Figura 20 apresenta as principais entidades de negócio do modelo SID (nível 1).

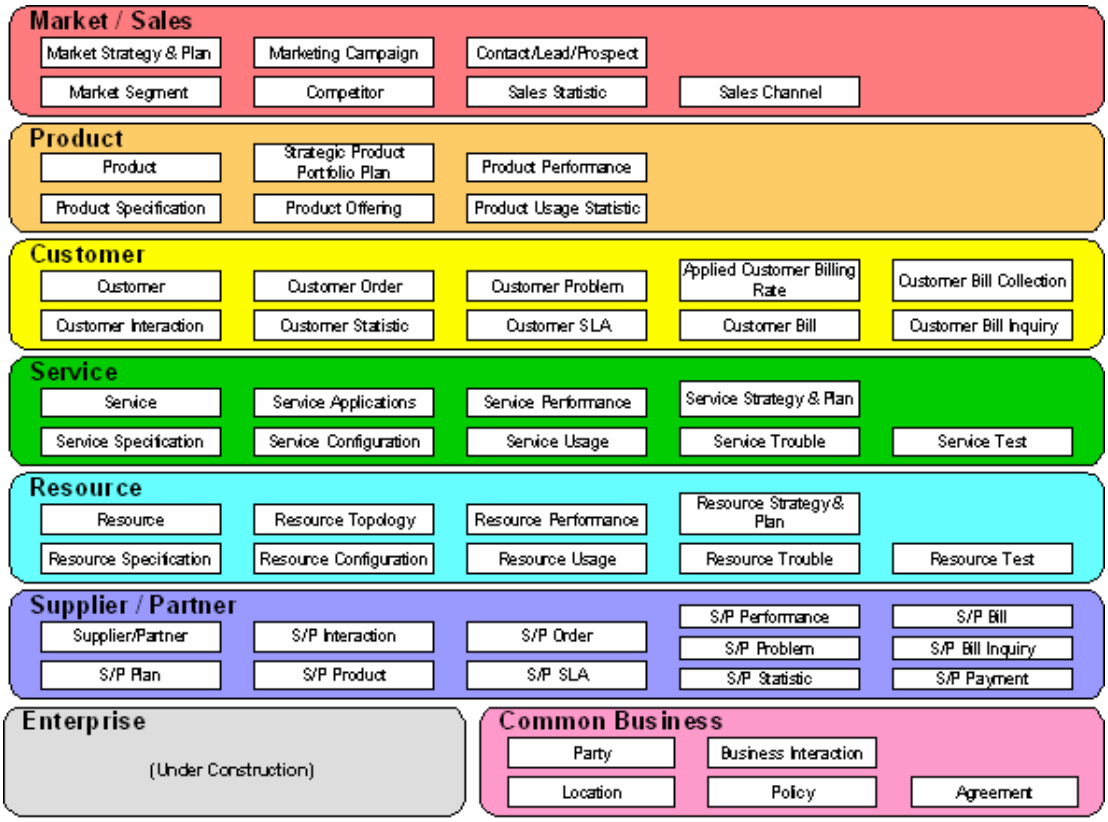


Figura 20 – Entidades do modelo SID [20]

A vista de negócio do SID está focada nas definições de entidade de negócio e nas definições dos atributos. Estas definições fornecem uma perspectiva orientada ao negócio da informação e dos dados. Quando combinados com o modelo de classes UML orientado ao negócio e esquemas XML, as definições fornecem uma vista de negócio dos dados e da informação.

A vista de negócio é organizada usando o *SID Model Framework*, que foi desenvolvido aplicando os conceitos de afinidade de dados aos processos e dados organizacionais, de modo a se ter uma vista em que a informação e os dados organizacionais não seja redundante. O resultado é uma framework dividida em camadas [Figura 20].

A vista de sistema estende a vista de negócio, adicionando detalhes acerca das entidades de negócio. Estes detalhes incluem atributos adicionais, operações e associações de classes como se pode ver na Figura 21.

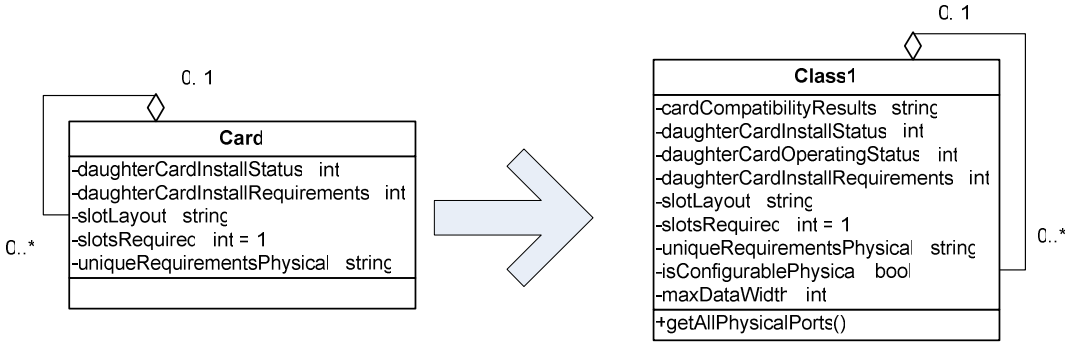


Figura 21 – Extensão para o modelo SID

Também existe a vista de implementação, que transforma o modelo de informação no modelo de dados. Actualmente a vista de implementação do SID está a ser desenvolvida pela iniciativa OSS *throug* Java (OSS/J).

2.6. OSS/J (OSS THROUGH JAVA)

A actual tecnologia OSS não consegue lidar com o aumento de escala das redes actuais, da diversidade de tecnologias e da diminuição do tempo para a apresentação de novos serviços [21], não esquecendo a disponibilidade e a fiabilidade.

Estas preocupações levaram ao aparecimento de uma nova aproximação para fornecer soluções OSS. A iniciativa OSS *through* Java (OSS/J) [22] define um conjunto de especificações de APIs independentes da tecnologia, de forma a implementar o programa NGOSS.

Através do programa Java Community Process (JCP) são definidas as especificações das APIs, as implementações de referência e perfis multi-tecnologia (Java, XML e *WebServices*) de forma a integrar e explorar as soluções OSS. A especificação OSS/J usa o modelo Core Business Entities (CBE), que é baseado no modelo SID, fornecendo assim uma implementação independente da tecnologia na arquitectura NGOSS.

O OSS/J inclui, entre outras, APIs que suportam gestão de tarefas (*Order Management*), activação de serviços (*Service Activation*), testes (*Testing*), tarifação (*Billing*), monitorização de desempenho (*Performance Monitoring*), inventários de recursos e de serviços (*Resource/Service Inventory*) e descoberta de recursos e serviços (*Resource/Service Discovery*).

2.6.1. ARQUITECTURA

A arquitectura do OSS/J é composta pelos CBEs, a *Common* API e as APIs para cada funcionalidade a implementar. Os CBEs e a *Common* API formam o cerne que serve de base às APIs funcionais (*Order Manageent*, QoS, RI, SI,...). Cada uma das APIs fornece três tipos de perfis de interacção que servem de interface a clientes externos. Estes perfis são o JVT *SessionBean*, XML e *WebServices*.

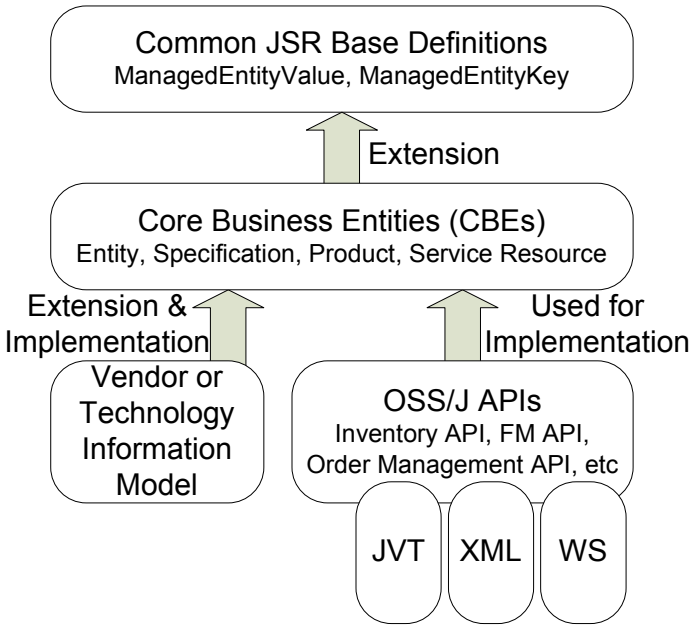


Figura 22 – Arquitectura OSS/J [21]

2.6.2. CORE BUSINESS ENTITIES

O modelo SID contém um conjunto de modelos e definições de *business entity* comuns que podem ser usados para a construção do core do modelo do OSS/J (CBEs) [23]. Este modelo pode ser estendido pelas diversas APIs do OSS/J, mas a estrutura mais básica não deve ser comprometida.

Business Entity

Caracterizada por atributos a *business entity* é parte do negócio. As *business entities* possuem outras propriedades, tais como associações, quer a outras entidades, quer a tipos de entidades, ou a exibição de comportamentos. Estas propriedades (atributos, associações e comportamento) das *business entities* podem ser herdados de outros tipos de entidades ou de outras *business entities*.

Core Business Entities

As CBEs representam cinco conceitos básicos, presentes no laborar de qualquer empresa: Quem (*Who*), o quê (*What*), onde (*Where*), quando (*When*) e porquê (*Why*).

As CBE podem ser um objecto que suporte um grupo de processos eTOM. Essas entidades suportam processos nas camadas de produto, serviço e recursos de eTOM. A CBE também pode ser um *business entity*, que é comum a um número de processos eTOM, como por exemplo *location*.

As CBE tanto podem representar uma abstracção como uma *superclasse* ou um objecto real.

Core Model

O *Core Model* contém os artefactos que descrevem completamente os CBE. Os artefactos incluem as definições dos CBE e modelos UML. Os modelos UML são compostos por diagramas de classes que documentam os atributos e operações das CBEs.

2.6.3. INVENTÁRIO

O OSS/J define três grupos de inventários o *Product Inventory*, o *Service Inventory* e o *Resource Inventory*, cada um destes grupos tem o seu conjunto de entidades e relações de inventário especificas. No entanto, todas as funções de inventário partilham abstracções comuns (entidades, associações e especificações de entidades) e um modelo base para interacção comum (*queries* baseadas em relações transversais, procedimentos atómicos de *update*, etc.) [24].

As funções de inventário são a parte central de uma solução integrada OSS. Fornecendo o armazenamento de recursos físicos, configurações, topologias de rede, recursos lógicos, serviços, produtos, etc. Também fornecem as operações de negócio necessários por outros componentes de OSS, por forma a pesquisar, monitorar, atribuir e actualizar a informação do inventário.

A API Inventário usa os seguintes conceitos para a criação do modelo:

- Entidade (*Entity*), são *Value Type Objects*, que representam conceitos de inventário tais como "Produto", "Serviço" e "Recurso"
- Especificações de Entidade (*Entity Specification*), são *Value Type Objects* representando especificações de entidades CBE.
- Associações (*Associations*), são *Value Type Objects* representando associações CBE ex. "`ResourceSupportsServiceAssocValue`".

Um diagrama de classe representando um modelo CBE do inventário pode conter os seguintes elementos:

- Interfaces representando Entidades;
- Interfaces representando Especificações;
- Associações, que podem ser classes associativas ou relações entre tabelas;
- *Constraints* (balizagem);
- Generalizações;

A Figura 23 seguinte mostra um exemplo com entidades e especificações, as classes estão definidas pelo respectivo estereótipo.

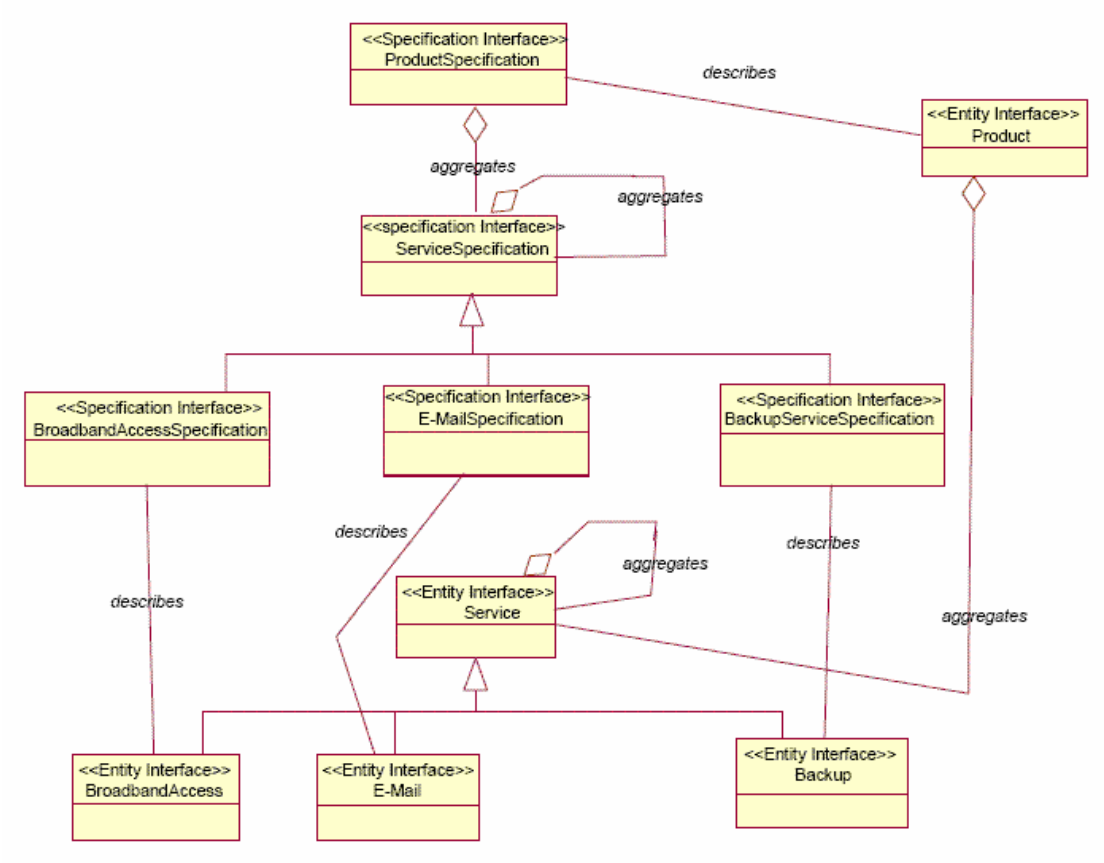


Figura 23 – Exemplo de entidades e especificações OSS/J [23]

2.7. SUMÁRIO

Neste capítulo foram apresentadas as definições das tecnologias usadas para a construção de uma aplicação de gestão de redes, nomeadamente a plataforma NGOSS e a implementação OSS/J. Foi feita uma síntese histórica da evolução das telecomunicações e dos sistemas de suporte às operações. Com o estudo destas tecnologias ficou perceptível a importância da criação de sistemas

independentes da tecnologia e a importância que actualmente os sistemas orientados ao serviço têm.

3. CASO DE ESTUDO

3.1. CONTEXTO

A PT Inovação está a construir uma nova geração de aplicações para a gestão de redes de próxima geração. Estas aplicações dividem-se pelas diferentes operações a efectuar sobre a rede, por exemplo o aprovisionamento de serviços fim a fim, a gestão de recursos, etc. Estas aplicações estão assentes sobre um modelo de informação genérico, baseado em normas internacionais. O módulo responsável pela cadastração dos serviços e dos recursos é o inventário.

O modelo de informação é especificado pelo modelo SID, a sua implementação usa as entidades CBE do OSS/J, sendo depois estendidas para as entidades PTIN. A criação do modelo de dados é feita usando a persistência dos *entity beans* do J2EE.

As entidades do modelo de informação são em primeiro lugar moduladas em UML e depois, usando a framework *openArchitectureWare*, são geradas as entidades.

O módulo de inventário é baseado na API *Inventory* do OSS/J e terá de ser o mais independente possível, de modo a poder ser usado por outros grupos, com a simples adição de um ficheiro JAR com as entidades a ser inventariadas.

Como exemplo deste caso de estudo será usada a aplicação *Site Manager*, que é a responsável pela criação da estrutura de localização dos recursos. Um recurso encontra-se numa dada localização (*Site*), esta localização pertence a uma dada área (*Geographical Area*), que por sua vez pertence a uma região (*Geographical Area*) e assim sucessivamente.

3.2. DESENHO DO SISTEMA

3.2.1. ARQUITECTURA

A arquitectura do módulo é baseada na API Inventário do OSS/J. O core do Inventário é o *SessionBean JVTInventorySessionBean*, que vai ter todas as funções necessárias para interagir com as entidades do inventário, ao mesmo tempo, por cada operação que é realizada é enviado um evento para o *Event Topic*, de modo a se poder fazer (entre outras operações) *log* sobre todas as interacções com o inventário.

A comunicação com a Base de Dados é feita usando *EntityBeans*, que se encarregarão de criar as tabelas, fazer os *queries*, *updates* e remoções.

As classes PTIn, que contêm a lógica de negócio usada na aplicação, estendem os CBEs do OSS/J. Estas classes são usadas pelos *EntityBeans* para a criação de tabelas e são usadas pelo *JVTInventorySessionBean* e pelo Cliente [Figura 24].

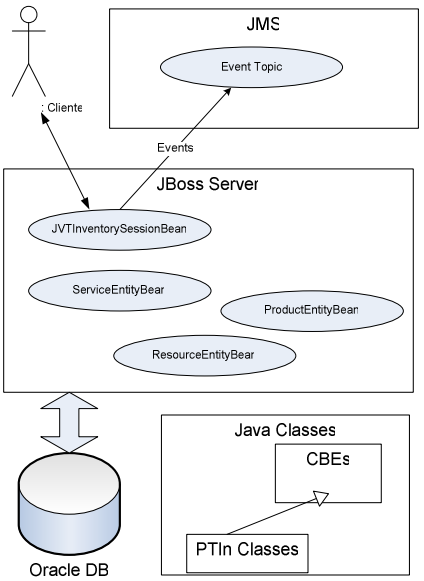


Figura 24 - Arquitectura

3.2.2. MODELO DE INFORMAÇÃO

O modelo de informação do *Site Manager* estende a *package location* da API *Common* do OSS/J, sendo composto por três entidades: a *GeographicLocationValue*, a *GeographicalAreaValue* e a *SiteValue*. Cada uma destas entidades possui uma *key* e um *specificationValue* e um *characteristicValue*. A Figura 25 mostra a relação entre as interfaces.

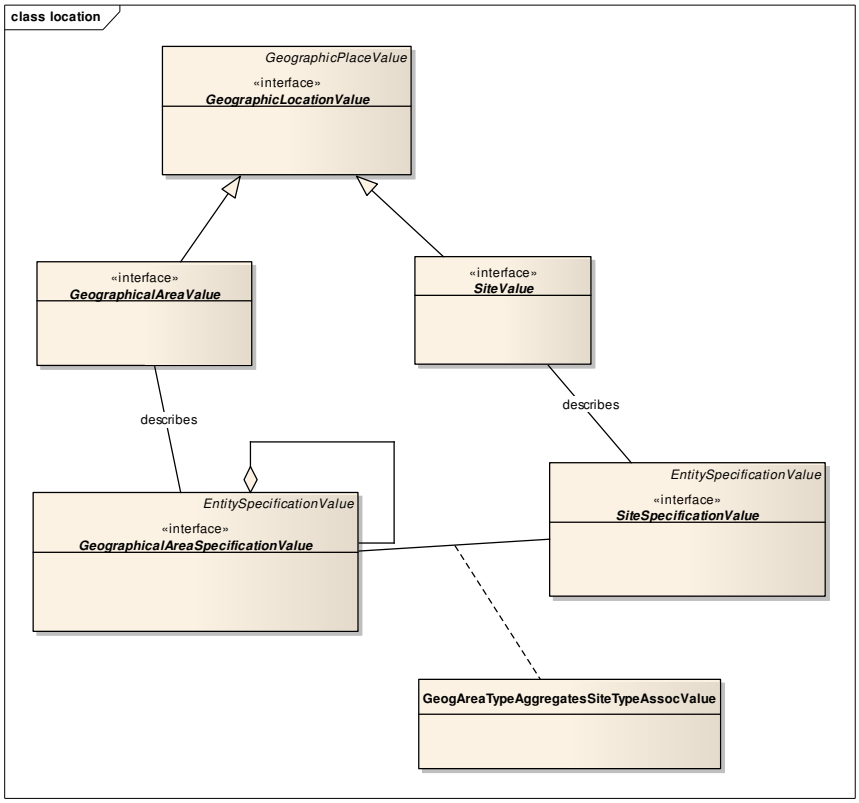


Figura 25 – Modelo de Informação do Site Manager[3]

De forma a tornar as entidades mais flexíveis, foram criadas cópias das entidades ao nível do inventário com os métodos necessários a interagir com o inventário. Na Figura 26 podem ver-se as relações entre as entidades, que são semelhantes às relações das entidades comuns. No entanto, podem-lhes ser

[3] O modelo completo encontra-se na Figura 35 do Anexo

acrescentados comportamentos e atributos que nada tenham a ver com as entidades comuns.

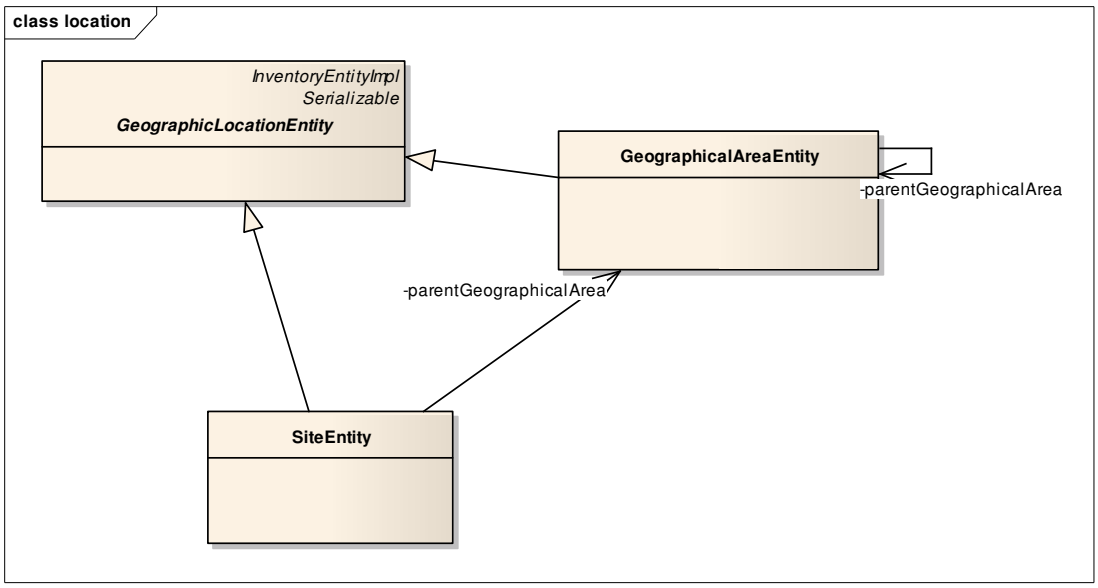


Figura 26 – Modelo de Informação das Entidades ao nível do Inventário[4]

3.3. IMPLEMENTAÇÃO

3.3.1. API INVENTÁRIO

A API Inventário do OSS/J é composta pelo *package ejb* (Figura 27), que contém as seguintes interfaces:

- *JVTInventorySession*, interface que apresenta as operações que podem ser feitas com o inventário, todas as operações são relativas às *Managed Entities* do modelo CBE.
- *Inventory*, esta interface apresenta todas as operações que as entidades ao nível do inventário têm de ter.

[4] O modelo completo encontra-se na Figura 36 do Anexo

- *InventoryCharacteristicSpecification*, esta interface apresenta todas as operações que as *SpecificationCharacteristicEntity* têm de implementar.

- *InventoryCharacteristicValue*, esta interface apresenta as operações que as *CharacteristicValueEntity* têm de implementar.

- *InventoryEntity*, estende a interface *Inventory* e apresenta as operações que as *Entity* têm de implementar.

- *InventoryEntitySpecification*, tal como a interface *InventoryEntity*, esta interface também estende *Inventory* e apresenta as operações que as *SpecificationEntity* terão de implementar.

Figura 27 – Packages usadas na API Inventário e operações da interface Inventory

Estas interfaces são implementadas por classes abstractas, que fornecem as operações genéricas. As operações que requerem uma implementação mais

particular e dependente da classe em questão, serão implementadas nas entidades – esta relação pode ser observada na Figura 28.

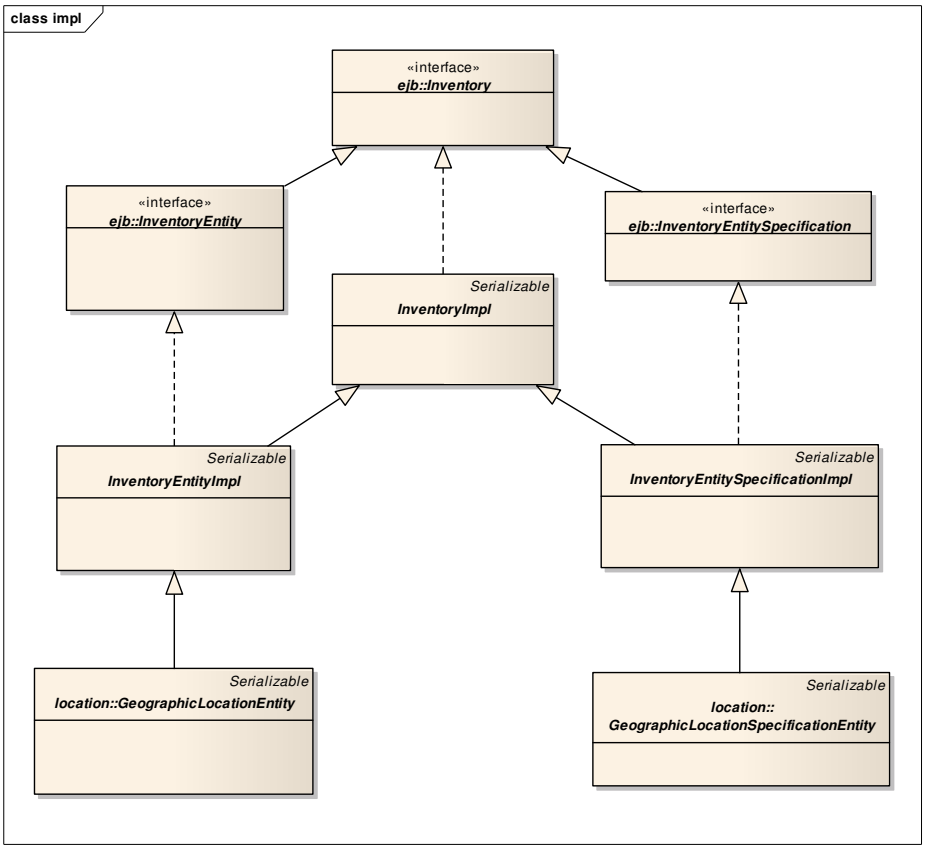


Figura 28 – Relação entre as interfaces da API do Inventário e as entidades

A *package factory* contém as classes:

- *InventoryEntityFactory* que é uma *factory* para construir as entidades ao nível do inventário a partir das entidades comuns. Esta *factory* é usada pelo *JVTInventorySessionBean*.
- *InventoryLookup*, faz o *lookup* do *JVTInventorySessionBean*. Esta classe é usada pelo cliente do inventário para instanciar o *JVTInventorySessionBean*.

A Figura 29 mostra a criação de uma entidade.

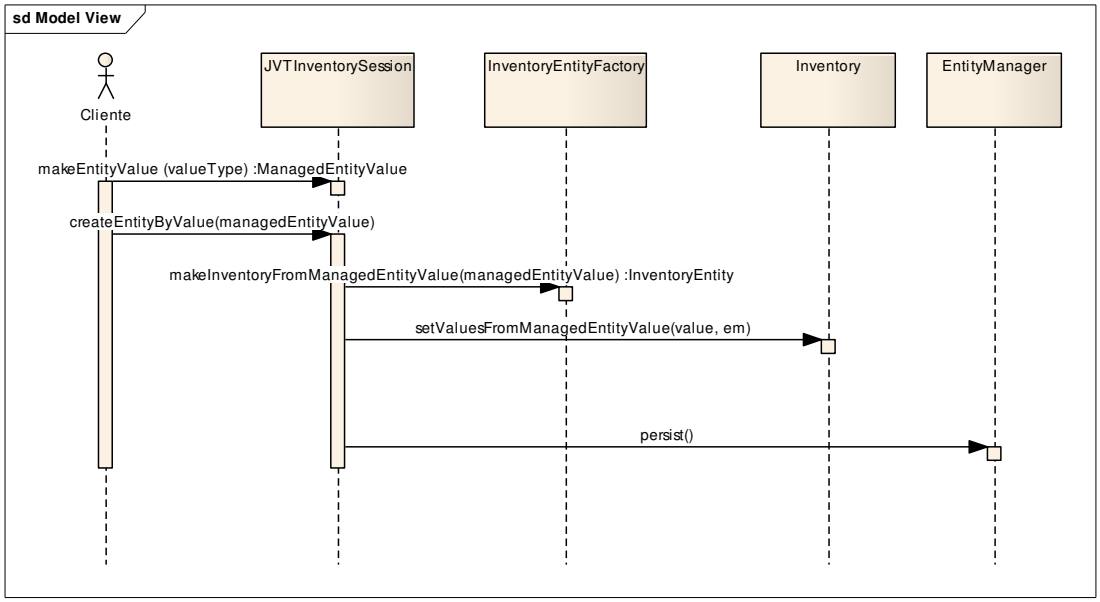


Figura 29 – Criação de uma entidade[5]

Observando o diagrama de sequência, é simples perceber-se o funcionamento do sistema, a aplicação cliente, antes de efectuar qualquer operação sobre uma entidade tem de pedir ao *JVTInventorySessionBean* para instanciar essa entidade, esta operação é feita com o *makeEntityValue*, o parâmetro a passar é uma *String* com o *full qualified name* da entidade. Criada a entidade o cliente tem de só preencher os atributos. Para colocar a entidade na base de dados, o cliente chama o método *createEntityByValue*, o *JVTInventorySessionBean* instancia a *InventoryEntity* e faz o *set* dos atributos no respectivo *Entity Bean* (*setValueFromManagedEntityValue*). No final o *JVTInventorySessionBean* chama o *persist* do *EntityManager* para fazer o *commit* dos valores na base de dados.

Na Figura 30 pode-se ver a base de dados criada e as entradas na tabela GEOGRAPHICAL_AREAS.

[5] Em anexo (Figura 37) encontra-se o diagrama de sequência de uma criação, leitura e remoção de uma entidade

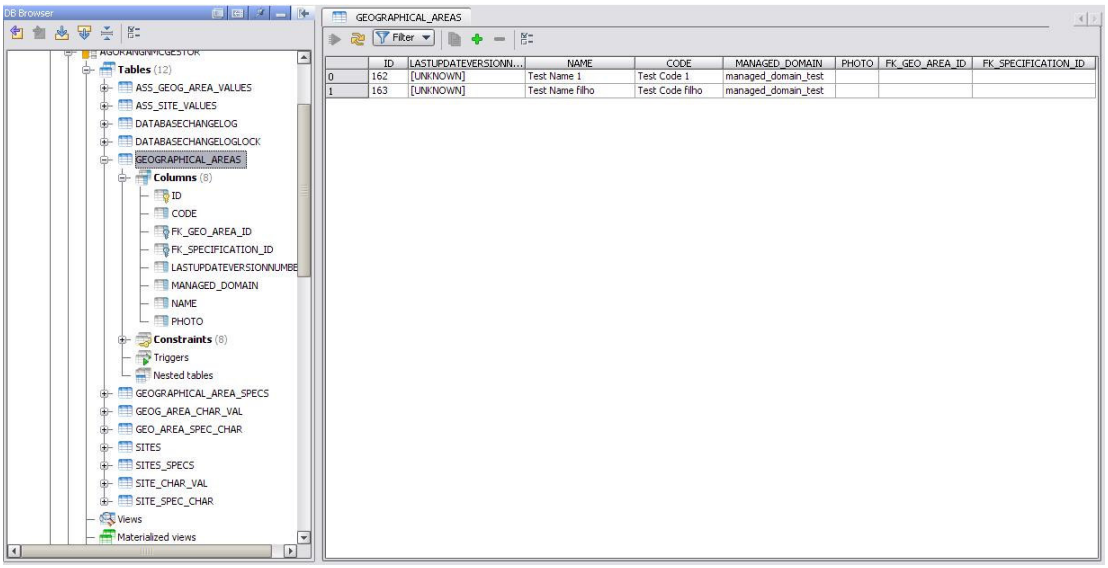


Figura 30 - Base de Dados

3.3.2. MODELAÇÃO

Com base no diagrama de classes UML, vão-se gerar as entidades usando a aplicação *openArchitectureware* (oAW) [25]. Para facilitar a operação usou-se o *plugin* UML2Exporter, que pega no projecto criado pelo *Enterprise Architect* e cria um modelo novo UML que o oAW possa interpretar.

A aplicação oAW possui uma linguagem de *script*, chamada XPand, com a qual se cria um *template* para se gerar os ficheiros. O *template* Root.xpt contém a lógica que transforma o modelo UML na estrutura Java que se pretende.

O modelo UML que serve de base é o que se encontra na Figura 35, em Anexo, e pretendem-se gerar quer as entidades da API *common*, quer as entidades ao nível do inventário. A seguir ir-se-á descrever a geração das entidades *common*. A Figura 31 mostra uma parte do modelo UML que foi convertido pelo *plugin* UML2Exporter, que vai ajudar na compreensão do funcionamento do *template* Root.xpt.

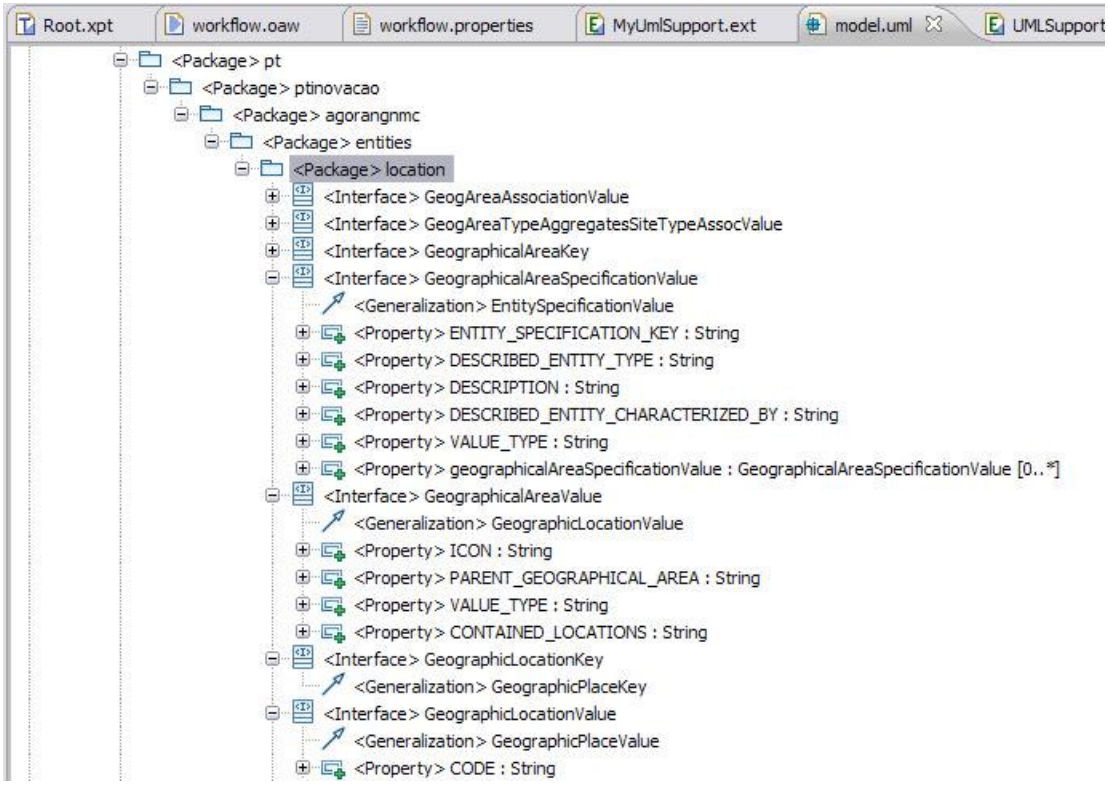


Figura 31 – Modelo UML do IDE Eclipse

O Root.xpt vai iterar todas as packages, recursivamente até encontrar uma interface ou classe (neste caso o modelo UML só se baseia em interfaces). Para cada interface que encontra vai criar dois ficheiros, a interface e a sua implementação. Depois itera todos os atributos e operações para assim acabar de preencher os ficheiros.

O pedaço de código seguinte, em XPand mostra o código que é necessário para gerar a interface de uma entidade:

```
«EXTENSION xtend::UMLSupport»
«EXTENSION xtend::MyUmlSupport»

«DEFINE Root FOR uml::Model»
      «EXPAND PackageRoot FOREACH ownedElement»
«ENDDEFINE»

«DEFINE PackageRoot FOR uml::Package»
      «EXPAND PackageLeaf FOREACH ownedElement»
```

```
«ENDDEFINE»

«DEFINE PackageLeaf FOR uml::Package»
      «EXPAND PackageLeaf FOREACH ownedElement»
      «EXPAND ClassRoot FOREACH ownedType»
«ENDDEFINE»

«DEFINE ClassRoot FOR uml::Interface»
      «FILE   interfacePathName() + "/" +name+".java"»
            package «interfacePackageName()»;
            public interface «name» extends «getGeneralizationName()»{

            «FOREACH attribute AS attr»
                  «IF attr.isStatic»
                        public «estatico(attr)» «attr.type.name»
«attr.name» = «attr.getDefault()»;
                  «ENDIF»
            «ENDFOREACH»

            «FOREACH getOperations() AS oper-»

                        «FOREACH oper.ownedComment AS comment-»
                        /**
                        «comment.body»
                        */
                        «ENDFOREACH-»

                  «oper.type.name» «oper.name» (
                  «FOREACH oper.ownedParameter AS el-» «el.name != null
? el.type.name + " " + el.name : ""» «el.name != null &&
oper.ownedParameter.last().name != el.name ? ",":""-»
                  «ENDFOREACH»)
                  «FOREACH oper.ownedComment AS comment-»
                        «comment.body.startsWith("@throw")?
comment.body.replaceFirst("@",""):""-»

                  «ENDFOREACH-»;

            «ENDFOREACH»

            }
      «ENDFILE»
«ENDDEFINE»

«DEFINE Root FOR uml::Element»«ENDDEFINE»
«DEFINE PackageRoot FOR uml::Element»«ENDDEFINE»
«DEFINE PackageLeaf FOR uml::Element»«ENDDEFINE»
«DEFINE ClassRoot FOR uml::Element»«ENDDEFINE»
```

O resultado deste *template* são as interfaces que estão definidas no módulo UML e pode "ser visto" na Figura 32 onde está apresentada a árvore do projecto oAW (os ficheiros gerados encontram-se na pasta src-gen).

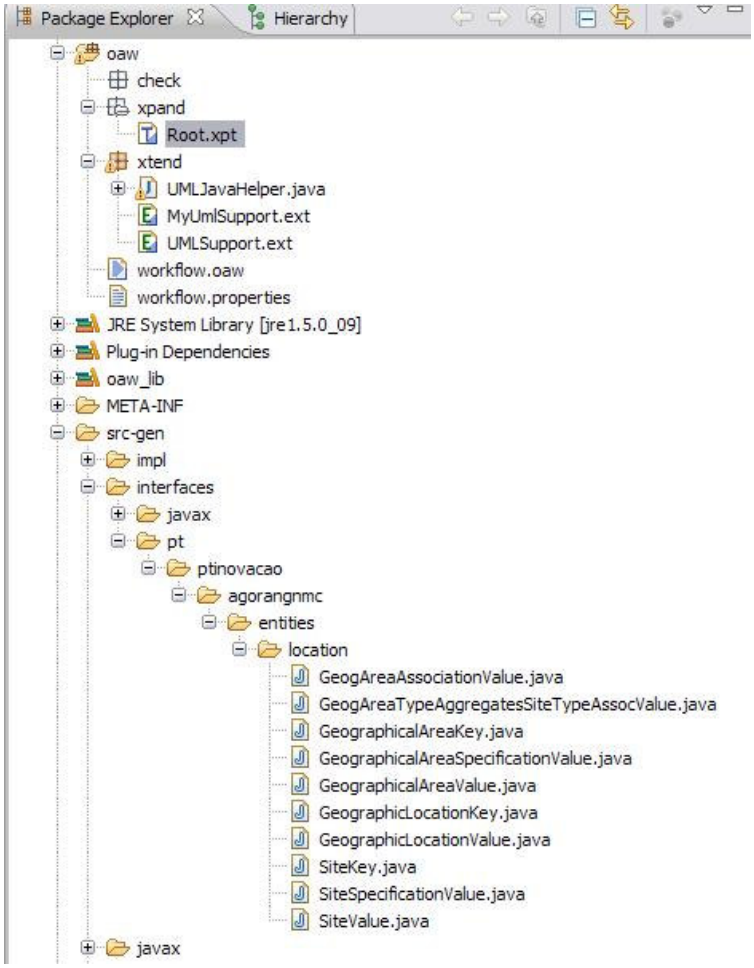


Figura 32 – Ficheiros gerados pelo oAW

3.4. FERRAMENTAS UTILIZADAS

3.4.1. JAVA ENTERPRISE EDITION 5

A *Java Enterprise Edition* é a tecnologia *standard* para a criação de aplicações Java portáveis, robustas, escaláveis e seguras do lado do servidor [26]. As aplicações *enterprise* são compostas por camadas, de forma a separar as partes da aplicação em pedaços independentes. Este tipo de modelo (multi-camada) é uma extensão do modelo de duas camadas, adicionando a camada do servidor aplicacional entre as camadas cliente e a camada da base de dados. Uma aplicação *enterprise* é composta pelas seguintes camadas [27]:

- Cliente
- Apresentação
- Negócio
- *Enterprise Information System* (EIS)

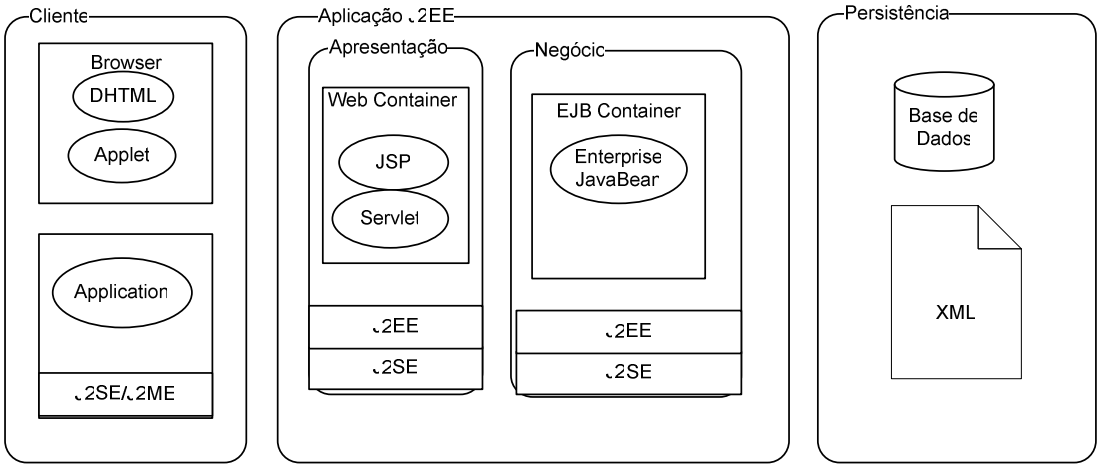


Figura 33 – Camadas de uma aplicação J2EE

Os componentes da camada cliente correm na máquina do cliente, as camadas de apresentação e de negócio estão a correr no servidor aplicacional Java EE, como por exemplo o JBoss, os componentes EIS correm no servidor de EIS. No caso do Inventário a camada de persistência é uma base de dados Oracle.

O Java EE consiste em especificações para implementar as camadas de apresentação e negócio duma aplicação. Essas especificações incluem Java *Servlets*, *JavaServer Pages* e *Enterprise Java Beans*. O Java EE fornece um modo *standard* para a interligação entre todos estas tecnologias, usando o *Java Naming and Directory Interface Enterprise Naming Context* (JNDI ENC), assim sendo diferentes componentes podem fazer *look up* e injectar recursos. O XML é usado para os *deployment descriptors* usados pelo servidor aplicacional.

EJB 3.0

A arquitectura *Enterprise JavaBeans* (EJB) é a arquitectura para o desenvolvimento e exploração de aplicações de baseadas em componentes. Os EJBs existem dentro do *container* EJB que é parte do servidor de aplicações. O *container* EJB é responsável pelo ciclo de vida de um EJB, pelo *pooling* de instanciação, pelo *naming* e serviço de directoria, gestão de transacções, serviço de *messaging* e manipulação de persistência. Na versão 3.0 a tecnologia EJB foi completamente remodelada de modo a simplificar o processo de desenvolvimento e tornar a tecnologia mais intuitiva. Agora os EJBs podem usar *annotations*, que fazem parte do Java desde a versão 5.0 [28], tornando desnecessários os *deployment descriptors*. Também deixou de ser necessário a implementação dos métodos da interface EJB 2 como o *create()* [29].

Java persistence e Entity Beans

A partir do EJB 3.0, a persistência passou a ter uma especificação própria, chamada *Java Persistence* 1.0 API e deixou de fazer parte da especificação EJB. Com o *Java Persistence* é possível mapear automaticamente objectos Java numa base de dados relacional e o inverso.

Os *Entity Beans* são simples objectos Java (POJO – *Plain Old Java Objects*) que podem ser serializados e instanciados com o operador *new*(). Tal como as entidades de uma base de dados relacional, os *entity beans* descrevem o estado e o comportamento de objectos reais. Contêm propriedades que com os métodos *getter* e *setter* correspondem às colunas da entidade que representam. Fornecem um modo simples e reutilizável de aceder à base de dados. A criação, remoção e alteração de um *entity bean* e as *queries* à base de dados são realizados pelo entity manager. O *entity manager* é responsável pelos mapeamentos Objecto/Relacional. Os EJBs só se tornam persistentes quando interagem com o *entity manager*. As entidades podem ser *managed* ou *unmanaged*, em que *managed* significa que o *entity*

bean está ligado ao *entity manager* e o seu estado está síncrono com a base de dados. Uma entidade fica *unmanaged* quando é devolvida ao cliente sob forma de valor de retorno. O contexto de persistência existe enquanto existe a transacção. Agora que os *entity beans* podem ser retornados aos clientes os *value objects* deixam de ser necessários [30].

EJB-QL

O EJB-QL é uma linguagem de query para comunicar com bases de dados relacionais, muito similar ao SQL, mas desenvolvida especificamente para interagir com objectos Java, sendo por isso independente da base de dados usada. O EJB-QL não interage directamente com tabelas e relações, mas sim com as propriedades das *entity beans* e com as suas relações. Usa o esquema de persistência das entidades para identificar os *beans* e as propriedades relação e assim navegar pelas relações. O *entity manager* é responsável por traduzir a *query* em SQL nativo da base de dados. O SQL é então executado usando o *driver JDBC* [27].

Session Beans

Os *Session Beans* são usados para implementar o fluxo de uma aplicação. Inclui a lógica de negócio, trata os dados da base de dados, alterar esses dados, quer através de entity beans quer através de acesso directo através do *entity manager*. Existem dois tipos de *session beans* (*stateful* e *stateless*), diferindo apenas no ciclo de vida e no modo como mantêm uma conversação com o cliente [27].

Stateless Session Beans

Os *stateless session beans* não mantêm o estado entre as chamadas aos seus métodos, são usados para pedidos que só necessitem de um método para realizar a operação pretendida. Cada método terá de ser autónomo e representar um serviço que pode ou não retornar um valor ao cliente. Um *stateless session bean* pode ser atribuído a um novo cliente à medida que os seus métodos forem invocados. O seu ciclo de vida consiste em dois estados, não existe ou está no *method-ready pool*, que indica que pode ser atribuído a qualquer cliente que necessite de um serviço desse *bean* [27].

Stateful Session Beans

Ao contrário dos anteriores, os *stateful session bean* mantêm um estado de conversação com o cliente estando dedicados a esse cliente durante todo o ciclo. Guardam dados que podem ser usados por todos os métodos e durante várias chamadas a métodos, por isso um método pode ser dependente do anterior. Um *stateful session bean* pode ter três estados: não existe, *ready* e *passive*. No estado *ready* os métodos podem ser invocados pelo cliente. O *stateful session bean* passa para o estado *passive* quando durante um longo período de tempo o cliente não faz qualquer pedido. Quando o cliente faz um pedido o estado é activado. Este processo pode repetir várias vezes durante a vida do *session bean*. Durante estas transacções o estado da conversação terá de ser guardado para futuras chamadas aos métodos [27].

JMS e Message Driven Beans

JavaMessage Service (JMS) é a *api* responsável por enviar mensagens assíncronas. Para o envio de mensagens é necessária uma ligação ao servidor de JMS e um endereço de destino. As mensagens são enviadas para *topics* ou *queues*,

dependendo do objectivo da mensagem. O cliente de JMS que envia a mensagem é chamado de *producer* e quem recebe a mensagem é chamado de *consumer*. Todos os EJBs podem enviar mensagens usando este serviço. Ao se enviar as mensagens, tem de ser ter em conta que os contextos de segurança e de transacção não são propagados do *producer* para o *consumer*.

Os *Message Driven Beans* (MDBs) são componentes usados para o processamento de mensagens assíncronas provenientes do JMS. Estes componentes são transaccionáveis, *stateless* e existem do lado do servidor. Os MDBs diferem dos outros EJBs, porque não possuem interfaces locais ou remotos, o cliente não pode interagir directamente com o MDB. Como a execução do MDB é completamente desacoplada do cliente, este não necessita de esperar pelo fim do pedido. Os MDBs podem ser usados para tarefas que não necessitem de valores de retorno e quando a execução da mensagem leve bastante tempo a se realizar [27].

3.4.2. FRAMEWORK PARA GERAR ENTIDADES

A arquitectura baseada em modelos (MDA – *Model Driven Architecture*) separa a lógica de negócio da tecnologia usada para a criação de uma aplicação. Usando a linguagem UML, podem-se criar modelos independentes da tecnologia, de funções e comportamentos da aplicação. Estes modelos podem então ser realizados em qualquer tecnologia, usando a MDA [31].

O openArchitectureWare (oAW) [25] é uma framework *open source* de MDA, implementada em JAVA. Suporta o *parsing* de modelos arbitrários e uma família de linguagens para validar e transformar esses modelos e gerar código baseado nesses modelos. O oAW é um plugin do IDE Eclipse, usando por isso para construção dos modelos a EMF (Eclipse *Modeling Framework*), mas também suporta outros tipos de modelos, como UML2, XML ou *JavaBeans*.

O core da aplicação é um motor de fluxo de trabalho, que permite a definição de fluxos de geração ou transformação. Pode ser usado um variado número de fluxos já existentes, para ler, e instanciar modelos, validá-los, transformá-los noutros modelos e finalmente, gerar código.

3.4.3. AMBIENTE DE PROGRAMAÇÃO

IntelliJ IDEA e Eclipse IDE

O recurso a ferramentas *Integrated Development Environment* (IDE), com o *syntax highlighting*, *auto completion*, *refactoring* e outras facilidades acelera o processo de desenvolvimento.

O IntelliJ IDEA é um IDE comercial da Jetbrains [32]. O editor, centrado em código é caracterizado pelas funcionalidades de assistência, *smart code completion* e *refactoring*, o processo de "ajudas" do editor é não intrusivo. Em relação ao Eclipse, no IDEA a capacidade de detecção de alterações feitas por aplicações externas, quer aos ficheiros de código, quer às bibliotecas é mais rápida e mais fiável.

O Eclipse é um IDE open source e extensível através de plugins e tornou-se um IDE de referência para a linguagem Java. Actualmente o Eclipse fornece editores para outras linguagens, tais como C/C++, COBOL, entre outras [33].

Enterprise Architect

Para fazer a modulação em UML foi usada a aplicação Enterprise Architect. O Enterprise Architect é uma ferramenta de modulação intuitiva, suporta UML 2.1, permite importar e exportar os modelos em XML [34]. Apesar de ser uma aplicação comercial, apresenta uma boa relação funcionalidades/preço, comparando com aplicações concorrentes (Magic Draw, Rational Rose).

CVS

O *Concurrent Versions System* (CVS) [35], é um sistema controlador de versões criado em 1985, por Dick Grune, de forma a contornar as limitações do *Revision Control System* (RCS) [36]. O RCS só trabalhava com ficheiros únicos, enquanto que o CVS é capaz de organizar as configurações de software em repositórios, módulos, directorias e ficheiros.

O CVS introduz a noção de repositório que é organizado em módulos. Em cada módulo existem ficheiros de configuração, que também são controlados. O utilizador acede ao repositório de modo a fazer *check out* aos ficheiros, obtendo assim uma cópia que será guardada localmente, não havendo assim ficheiros partilhados.

As boas práticas da utilização do CVS aconselham à sincronização diária e o código a ser sincronizado terá de ser compilável, de forma a não introduzir erros nas versões. O fluxo de sincronismo é 1) *update* do módulo, 2) correcção de possíveis conflitos, 3) *commit* dos ficheiros alterados e 4) novamente *update*.

Apache Maven

Numa aplicação Java convencional, para se compilar a aplicação terão de se criar vários ficheiros de *build* do Ant, todos muito parecidos e de forma a se assegurar a consistência na compilação, os ficheiros *jar* teriam de ser colocados no CVS. O projecto Maven [37] vem deste modo criar uma forma *standard* de construir os projectos, criando uma definição clara sobre em que é que o projecto era composto, uma forma simples de se publicar a informação do projecto e uma forma de partilhar as bibliotecas *jar* através de vários projectos distintos. O objectivo principal do Maven é de auxiliar na construção, teste e documentação de

projectos Java de um modo simples e consistente durante o ciclo de vida do projecto.

A filosofia Maven é de só colocar em CVS o código fonte, sendo as bibliotecas guardadas num repositório chamado Maven Repository – uma localização remota onde os ficheiros são guardados por projecto e por versão – permitindo assim que a gestão de dependências seja feita de uma forma simplificada. Adicionalmente podem ser criados repositórios locais, ao nível do posto de trabalho quer ao nível corporativo.

O Maven usa, para guardar a informação do projecto, ficheiros descritores em XML, chamados Project Object Model. Nestes ficheiros encontra-se a informação relativa ao projecto, como o nome do projecto, descrição, quem desenvolve, etc.

JBoss

O *JBoss Application Server* (JBOSS AS) é uma plataforma J2EE *open source* para o desenvolvimento e deploy de aplicações *Java enterprise*, aplicações web e portais [38]. O JBoss AS é parte do *JBoss Enterprise Middleware System* (JEMS). JEMS é uma suite de produtos *middleware* extensível e escalável que contém produtos *middleware* completamente integrados e testados, tais como o JBoss AS, o Apache Tomcat, JBoss Cache, JBoss Messaging, Hibernate e o JBoss Eclipse IDE.

3.5. SUMÁRIO

Neste capítulo foi discutida a arquitectura da API do Inventário e da API Common e o seu funcionamento. Foi também apresentada a implementação destas APIs e o módulo que gera as entidades. No final do capítulo mostraram-se

as diferentes ferramentas que foram utilizadas para a realização deste caso de estudo.

4. CONCLUSÕES

O estado actual das telecomunicações levou a uma mudança no modo como os operadores de telecomunicações fornecem os seus serviços. Com o fim do monopólio os operadores de telecomunicações tiveram de se virar para a satisfação do cliente e para o fornecimento de novos serviços. Esta mudança de paradigma levou a uma necessidade de diminuição de custos na manutenção e gestão da rede e a um incremento de serviços disponibilizados. A variedade de tecnologias empregues e a necessidade de suportar sistemas legados leva a que a solução também envolva uma abstracção em relação às tecnologias empregues para entregar um serviço.

Os consórcios internacionais ITU-T e TMForum criaram padrões que tentam resolver estes problemas, nomeadamente o NGN e o NGOSS. Para as redes de Próxima Geração a norma ITU-T Y.2001 define os requisitos necessários para que uma rede seja considera uma NGN. O programa NGOSS é uma norma para a criação e desenvolvimento de componentes OSS/BSS. A framework eTOM define os processos que interagem com o cliente e suporte à operação, processos que interagem com a infra-estrutura, relacionados com serviços específicos. A framework permite o desenvolvimento de processos *end-to-end* de modo a satisfazer o cumprimento do serviço, e os requisitos de tributação. O modelo SID representa a vista lógica dos objectos de interesse (entidades), tais como cliente, localização e elemento de rede. Também representa as relações (associações) entre entidades, tais como uma rede está numa dada localização.

A implementação dos conceitos apresentados pelo programa NGOSS é definida pela iniciativa OSS through Java (OSS/J) que define um conjunto de especificações de APIs independentes da tecnologia. Entre essas APIs encontram-se as usadas nesta dissertação, o Inventário e o modelo de informação (*Common*).

O modelo de informação e o Inventário representam a base de uma aplicação de gestão de redes, todos os outros módulos usam estes componentes para a realização das suas operações. Necessitam das entidades criadas no modelo de informação e necessitam do módulo de inventário para a cadastração e persistência de recursos e serviços de rede.

No caso de estudo analisado, o exemplo apresentado é muito simples e serve para demonstrar a flexibilidade do modelo de informação e a capacidade de generalização existente no modelo definido pelo SID. O módulo de Inventário apresentado consegue ser genérico de modo a ser usado por outros grupos, desde que as suas entidades estendam as entidades CBEs da API *Common* do OSS/J, bastando para isso acrescentar as bibliotecas com as suas entidades e *EntityBeans* ao *classpath*. O Inventário encarrega-se de cadastrar as entidades e criar as tabelas na base de dados.

A simplicidade da solução esconde a quantidade de conceitos necessários para a construção do modelo de informação e do Inventário. Para a criação do modelo de informação é necessário conhecer o modelo de rede, o modelo SID e a API *Common* do OSS/J. A implementação do módulo de Inventário, necessita, além do conhecimento do modelo de informação, do conhecimento da API *Inventory* do OSS/J, de conceitos avançados de desenho da arquitectura do sistema e de conceitos avançados de programação em Java/J2EE.

A aplicação geradora de entidades, oAW (*openArchitectureWare*), revelou-se uma aplicação bastante flexível e poderosa. Apesar de ter uma curva de aprendizagem um pouco lenta, permite que se possam gerar entidades desde o início. A utilização do *plugin* UML2Exporter permitiu evitar o trabalho fastidioso de criação do meta modelo, usando directamente o ficheiro de projecto do Enterprise Architect, tornando o processo mais rápido.

As tecnologias apresentadas são relativamente recentes e encontram-se em constante aperfeiçoamento. Durante a conclusão desta dissertação, o TMForum

apresentou uma nova especificação chamada de TIP (TMForum Interface Program)[39], que é o resultado de um esforço para integrar as diferentes normas de implementação do OSS.

Antes de se continuar com o desenvolvimento dos módulos que vão servir de cliente ao Inventário, nomeadamente o Order Management, o Discovery e outros serviços, o trabalho futuro passa antes de mais por analisar esta nova especificação do TMForum e verificar se traz vantagens para a aplicação.

REFERÊNCIAS

1. Gonçalves, J., Cadime, R., Laranjeira, A., Aguiar, M., Mirones, V., *Modelos de Informação para Gestão de Redes de Transporte - Aplicação na Gestão de Redes Netb@nd*. Saber e Fazer Telecomunicações, 2006(4): p. 118 - 125.
2. Woods, D. *PBXes: CO Switches: Extended to the Enterprise*. Digital Convergence 2001 [cited 2008-01-31]; Available from: http://www.networkcomputing.com/1220/1220ws1.html.
3. Technologies, P. *SS7 Tutorial*. [cited 2008/06/01]; Available from: http://www.pt.com/tutorials/ss7/.
4. Doc. RNDr. Peter Mederly, C. *Introduction to Computer Networks*. 1997 [cited 2008/01/31]; Available from: http://hq.alert.sk/~mandos/fmfi-uk/Informatika/Distribuovane%20Systemy/knihy/ICN/ch2s5.htm#2.5.
5. Zakon, R.H. *Hobbes' Internet Timeline v8.2*. 2006 2006/11/1 [cited 2008/01/31];
   Available from: http://www.zakon.org/robert/internet/timeline/.
6. *Network Node Interface for the Synchronous Digital Hierarchy (SDH)*. ITU-T Rec. G.707, 2002.
7. *Link Capacity Adjustment Scheme (LCAS) for Virtual Concatenated Signals*. ITU-T Rec. G.7042/Y.1305, 2001.
8. E. Hernandez-Valencia, M.S., and Z. Zhu, *The generic framing procedure (GFP): an overview*. IEEE Commun. Mag., 2002. **40**: p. 63-71.
9. Green, H., Ghiggino, P., *An Overview of Key Technologies for the Next Generation Networks*, in *Optical Networks And Technologies*, Springer, Editor. 2004, Springer p. 31 - 43.
10. *General overview of NGN*. ITU-T Rec. Y.2001, 2004.
11. *Generic functional architecture of transport networks*. ITU-T Rec. G.805, 2000.
12. Wang, L., Lv, T., *The NG-OSS Evolution of Telecom Service Providers: From Network-Focused to Customers-Focused*. IFIP International Federation for Information Processing, 2007. **255**.

13. *Principles for a telecommunications management network.* ITU-T Rec. M.3010, 2000.
14. *TOM Business Process Framework*. 2000 2000/09/10 [cited 2008 2008/04/20]; Available from: http://www.tmforum.org/clickmap/tomv2-1/di39.htm.
15. *NGOSS*. 2008 [cited 2008/04/10]; Available from: http://www.tmforum.org/NGOSS/1911/home.html.
16. *NGOSS Release 6.0 Solution Suite Release Notes.* NGOSS Release 6.0, 2005.
17. Kelly, M.B., *The TeleMAnagement Forum's Enhanced Telecom Operations Map (eTOM).* Journal of Network and Systems Management, 2003. **11**(1): p. 109-119.
18. *Business Process Framework (eTOM)*. 2008 [cited 2008/04/11]; Available from: http://www.tmforum.org/browse.aspx?catID=1647.
19. *Information Management*. 2008 [cited 2008/02/10]; Available from: http://www.tmforum.org/InformationManagement/1684/home.html.
20. *Business View Concepts, Principles, and Domains.* GB922 - SID Concepts, 2008.
21. *TM Forum OSS/J Program - OSS/J Roadmap*. 2008.
22. *TM Forum OSS/J*. 2008 [cited 2008/05/15]; Available from: http://www.tmforum.org/ossj/.
23. Strassner, J., Sigley, W., Hepburn, H., et. al, *Core Business Entities Model White Paper.* 2004.
24. Gauthier, P., *OSS Inventory API 1.2- JSR 142*. 2007.
25. *openArchitectureWare*. 2008 [cited 2008/05/10]; Available from: http://www.openarchitectureware.org/.
26. *Java EE at a Glance*. 2008 [cited 2008/05/10]; Available from: http://java.sun.com/javaee/.
27. Jendrock, E., et al., *The Java EE 5 Tutorial*. 2007.
28. *Java SE at a Glance*. 2008 [cited 2008/05/10]; Available from: http://java.sun.com/javase/.
29. *Enterprise JavaBeans Technology*. 2008 [cited 2008/05/10]; Available from: http://java.sun.com/products/ejb/.

30. *Java Persistence API*. 2008 [cited 2008/05/10]; Available from: http://java.sun.com/javaee/technologies/persistence.jsp.
31. Strassner J., F.J., Huang J., Faurer C., Richardson T.,, *TMF White Paper on NGOSS and MDA*. TeleManagement Forum / Object Management Group, 2004.
32. *JetBrains - IntelliJ IDEA*. 2008 [cited 2008/01/20]; Available from: http://www.jetbrains.com/.
33. *Eclipse - an open development platform*. 2008 [cited 2008/03/01]; Available from: http://www.eclipse.org/.
34. *Sparx Systems - Enterprise Architect*. 2008 [cited 2008/01/05]; Available from: http://www.sparxsystems.com.au/.
35. *Concurrent Versions System* 2006 2006/12/03 [cited 2008/05/10]; Available from: http://www.nongnu.org/cvs/.
36. *Revision Control System*. 2007 2007/02/07 [cited 2008/05/10]; Available from: http://www.gnu.org/software/rcs/rcs.html.
37. *Maven*. 2008 [cited 2008/05/10];
Available from: http://maven.apache.org/.
38. *JBoss*. 2008 [cited 2008/05/10]; Available from: http://www.jboss.org/.
39. *TM Forum Interface Program*. 2008 [cited 2008/05/20]; Available from: http://www.tmforum.org/BestPracticesStandards/InterfaceProgram/5733/Home.html.

ANEXOS

DIAGRAMAS

Figura 34 – Ampliação do Nível 2 do modelo eTOM

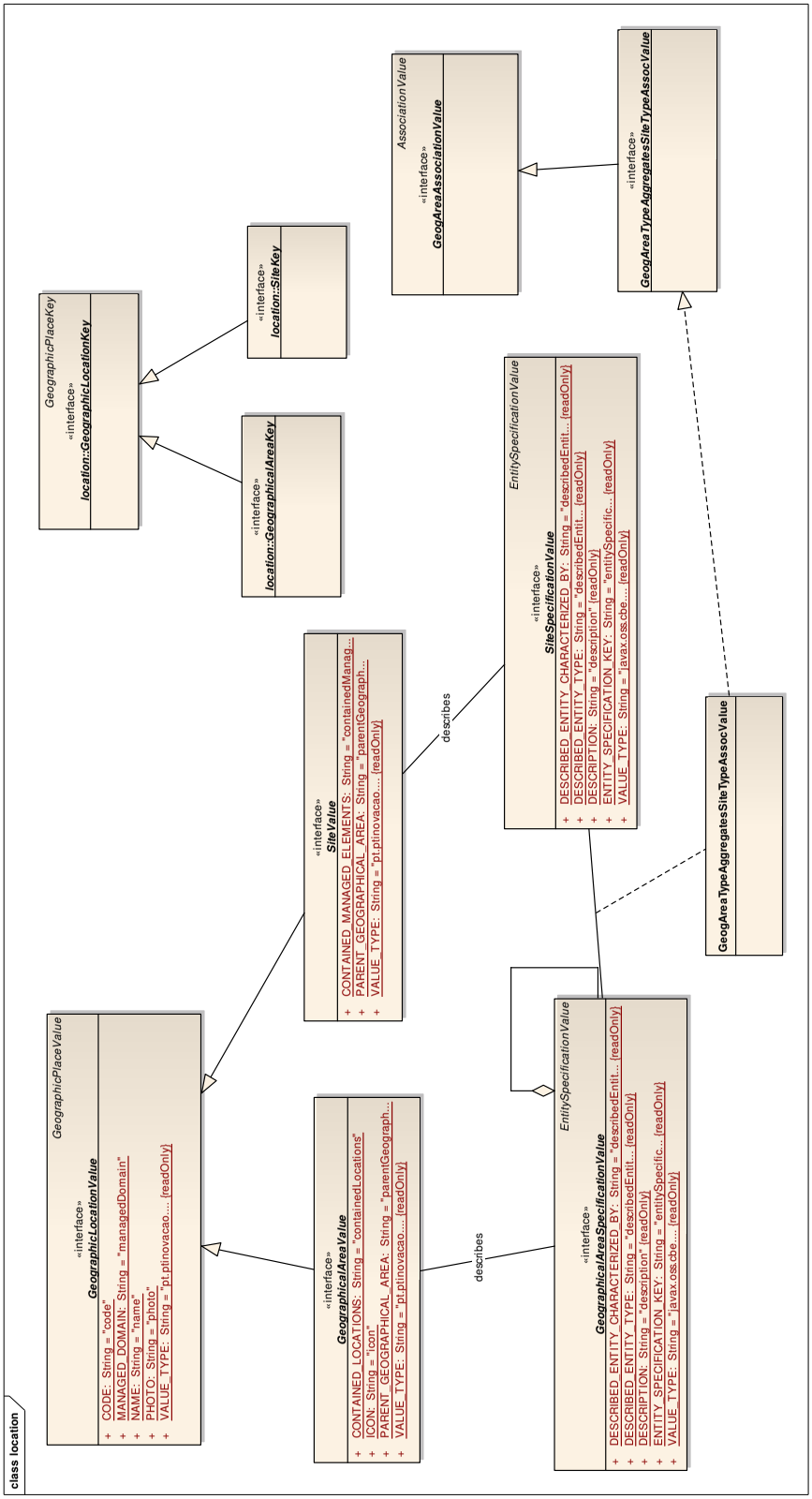


Figura 35 – Modelo de Informação do Site Manager

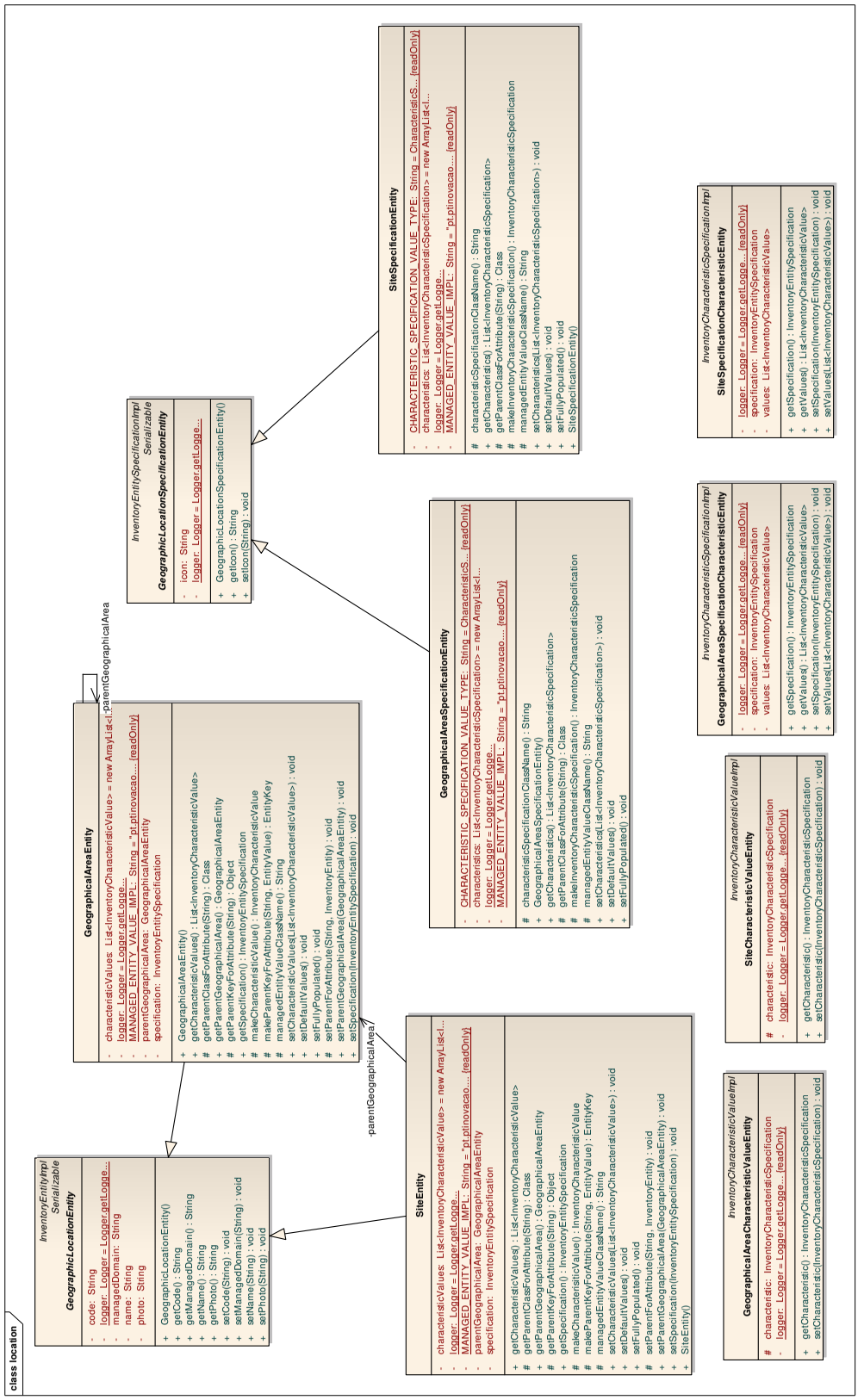


Figura 36 – Modelo de Informação das Entidades ao nível do Inventário

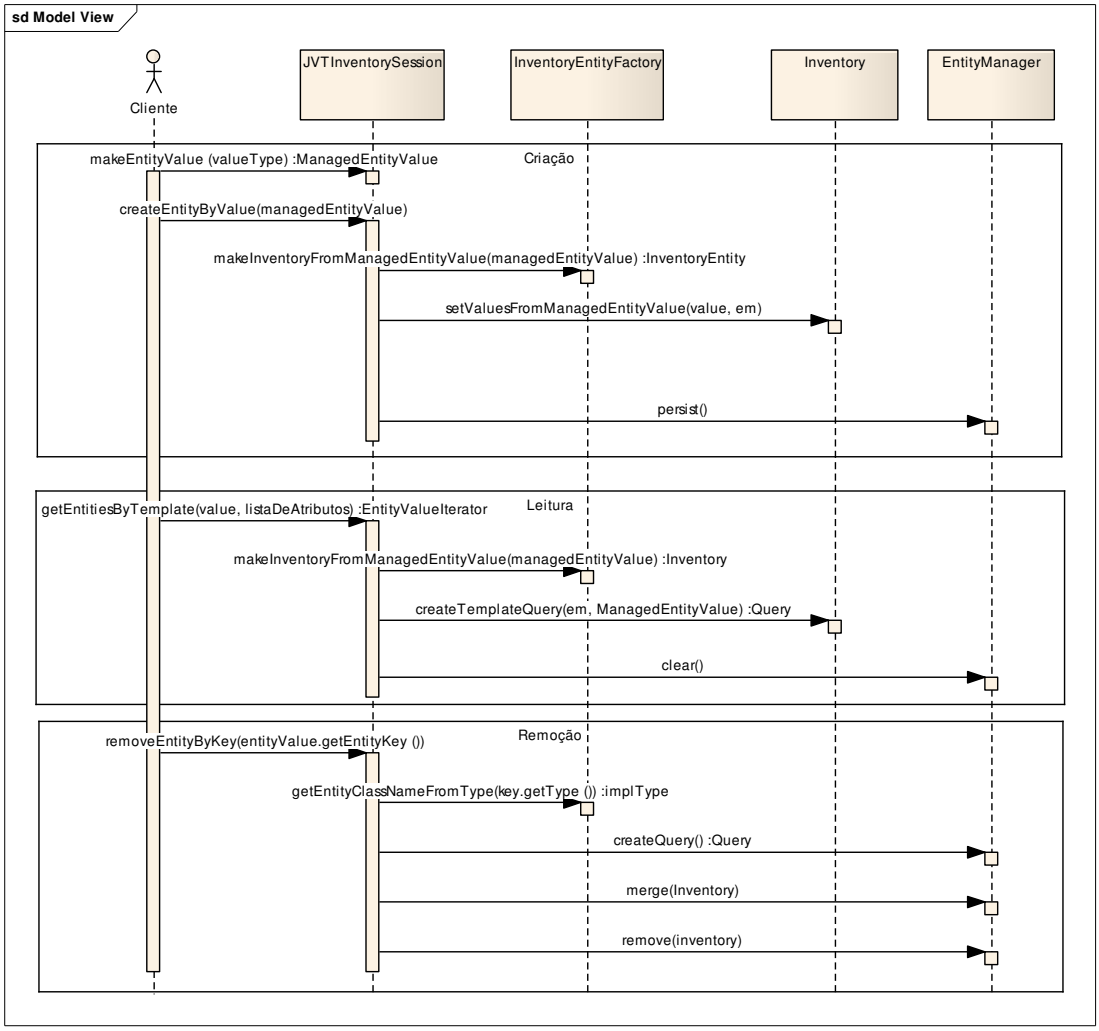


Figura 37 – Diagrama de Sequência das operações de Criação, Leitura e Escrita